robotron

Anwenderdokumentation

SYSTEMHANDBUCH

UDOS-1526

Systemhandbuch

(Anleitung für den Systemprogrammierer)
- für Bürocomputer A 5120/30
- für das Universelle Bildschirm-Terminal K 8931
- für das Programmiergerät PRG 700

mit dem Betriebssystem UDOS 1526

2. Auflage

VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt 1984

III-12-12 Kv 1127/84

Die vorliegende Dokumentation entspricht dem Stand vom 01.02.1984.
Zum System "UDOS 1526" gehören per 31.01.84 insgesamt folgende Dokumentationen:

- Systemhandbuch UDOS 1526
- Sprachbeschreibungen BASIC
  PASCAL
  FORTRAN
  (Z8ASM)
  PLZ/SYS
- System U 8000
- Beschreibung zum REASSEMBLER (REASM)
- Beschreibung zum Symbolischen Debugger (SYD)

Nachdruck, jegliche Vervielfältigungen dieser Unterlage oder Auszüge daraus sind unzulässig.

Die Ausarbeitung dieser Dokumentation erfolgte durch ein Kollektiv des VEB Robotron-Buchungsmaschinenwerk

Herausgeber:

VEB Robotron-Buchungsmaschinenwerk
Karl-Marx-Stadt
9010 Karl-Marx-Stadt
PSF 129

Die Entwickler/Herausgeber sind dankbar für Kritiken, Hinweise und Wünsche, die bitte schriftlich an den Bereich "Software-Zentrum" zu übergeben sind.

Inhaltsverzeichnis

		Seite
1.	Systemübersicht	1
1.1.	Bürocomputer A 5120/30 mit Betriebssystem UDOS 1526	1 - 2
1.2.	Programmiergerät PRG 700 mit UDOS 1526	3
1.3.	Struktur des Systems	4
2.	Hardware	5
2.1.	Übersicht Bürocomputer	5
2.2.	Bemerkungen zu den Komponenten	6
2.3.	Übersicht PRG 700	7
3.	Diskettenaufbau und -organisation	7 - 9
4.	Der ASC II - Code	9
5.	Grundaufbau des Betriebssystems	10 - 11
6.	Physisches Betriebssystem	11
6.1.	Debuggerkommandos	11 - 14
6.2.	Systemdaten	14 - 15
6.3.	Physische Programmodule	16 - 17
7.	UDOS-Executive und UDOS-Kommandos	18
7.1.	UDOS-Executive (OS)	18
7.1.1.	Ladevorgang und Initialisierung des Systems	18 - 19
7.1.2.	Speicherverwalter (MEMMGR)	19 - 20
7.1.3.	E/A-Struktur	20 - 22
7.1.4.	Kommandointerpreter	22
7.2.	Beschreibung der UDOS-Kommandos	23 - 25
7.2.1.	ACTIVATE	26
7.2.2.	ALLOCATE	26
7.2.3.	BRIEF	26
7.2.4.	CAT	27 - 28
7.2.5.	CLOSE	28
7.2.6.	COMPARE	28
7.2.7.	COPY	29

7.2.8.	COPY DISK	30
7.2.9.	DATE	30
7.2.10.	DEACTIVATE	31
7.2.11.	DEALLOCATE	31
7.2.12.	DEBUG	32 - 33
7.2.13.	DEFINE	34
7.2.14.	DELETE	35
7.2.15.	DISPLAY	35
7.2.16.	DO	35
7.2.17.	DUMP	36
7.2.18.	ECHO	36
7.2.19.	EXTRACT	36
7.2.20.	FORCE	37
7.2.21.	FORMAT	37
7.2.22.	HELP	38
7.2.23.	IMAGE	38
7.2.24.	INITIALIZE	39
7.2.25.	LADT	39
7.2.26.	MASTER	39
7.2.27.	MOVE	40
7.2.28.	PAUSE	41
7.2.29.	PRINT	41
7.2.30.	RELEASE	42
7.2.31.	RENAME	42
7.2.32.	RESTOR_TABS	42
7.2.33.	SAVE_TABS	43
7.2.34.	SET	43 - 44
7.2.35.	STATUS	44
7.2.36.	VERBOSE	45
7.2.37.	XEQ	45
7.2.38.	:	45
7.2.39.	HELP ASCII	45
7.2.40.	CHAR	45
7.2.41.	ERROR	45

8.	Das UDOS-Dienstsystem	46
8.1.	Dateiverwaltung Floppy-Disk	46
8.1.1.	Dateiname, Dateityp, Merkmale	46 - 48
8.1.2.	Floppy-Disk Laufwerkzuweisung	48
8.1.3.	UDOS - File - Debugger	48 - 51
8.2.	ZDOS	51
8.2.1.	Allgemeines	51 - 52
8.2.2.	Diskettenbelegung	52
8.2.3.	Satzaufbau und Datenorganisation	52 - 53
8.2.3.1.	Aufbau DESCRIPTOR	53 - 54
8.2.3.2.	Datei-Zeiger	54
8.2.4.	Aufrufkonventionen und Kommando-parameter	54 - 56
8.2.5.	Rückmeldung	56
8.2.5.1.	Allgemeine Meldungen	56
8.2.5.2.	Fehlermeldungen	56 - 57
8.2.6.	Aufrufe	58
8.2.6.1.	INITIALIZE	58
8.2.6.2.	ASSIGN	58 - 59
8.2.6.3.	OPEN	59 - 65
8.2.6.4.	CLOSE	65 - 66
8.2.6.5.	REWIND	66 - 67
8.2.6.6.	READ BINARY	67 - 68
8.2.6.7.	WRITE BINARY	68 - 69
8.2.6.8.	WRITE CURRENT	69 - 70
8.2.6.9.	DELETE	70 - 71
8.2.6.10.	DELETE REMAINING RECORDS	71
8.2.6.11.	ERASE	72 - 73
8.2.6.12.	READ AND DELETE	73 - 74
8.2.6.13.	READ CURRENT	74 - 75
8.2.6.14.	READ PREVIOUS	75 - 76
8.2.6.15.	READ DIRECT	76 - 77
8.2.6.16.	SKIP FORWARD	77 - 78
8.2.6.17.	SKIP BACKWARD	78
8.2.6.18.	SKIP TO END	79
8.2.6.19.	RENAME	79 - 80

8.2.6.20.	UPDATE	81
8.2.6.21.	SET ATTRIBUTES	82
8.2.6.22.	QUERY ATTRIBUTES	82 - 83
8.2.7.	Programmierung und Nutzung zusätzlicher Gerätetreiber	84
8.2.7.1.	Programmierung	84
8.2.7.2.	Programmbeispiel anhand des PRINTER-Treibers	85 - 88
8.2.7.3.	Nutzung allgemein	89
8.2.7.3.1.	Druckertreiber SD	90
8.2.7.3.2.	Lochbandtreiber PTAPE	91 - 92
8.3.	EDITOR	93 - 94
8.3.1.	Erfassen von Dateien	94
8.3.2.	Ändern von Dateien	94 - 95
8.3.3.	Kommandos des Editors	95 - 99
8.4.	ASSEMBLER	100
8.4.1.	Assemblerbedienung	100 - 102
8.4.2.	Sprachbeschreibung	102
8.4.2.1.	Ausdrücke	102 - 104
8.4.2.2.	Pseudobefehle	104 - 105
8.4.2.3.	Makros	105 - 109
8.4.2.4.	Assembler-Kommandos	110
8.4.2.5.	Abweichungen der Mnemoniks des UDOS-Assemblers von CABS 1520	110 - 111
8.5.	LINKER	112
8.5.1.	Funktion	112
8.5.2.	Anwendung des Linkers	112 - 115
8.5.3.	Optionen	115 - 119
8.5.4.	Listenformat	119 - 120
8.5.5.	Überlagerungen	120 - 121
Anlage 1	Besonderheiten PRG 700 mit UDOS 1526	122 - 124

1. Systemübersicht

1.1. Bürocomputer A 5120/30 mit Betriebssystem UDOS 1526

Die Bürocomputer A 5120 und A 5130 (bzw. die abgeleiteten, als gleichwertig betrachtbaren Terminals) wurden gemeinsam mit dem Betriebssystem SIOS 1526 entwickelt.

Als technisches Mittel dazu stand das Mikrorechnerentwicklungssystem MRES 20 zur Verfügung.

Die Bürocomputer /SIOS wurden für folgende Haupteinsatzgebiete entwickelt:

- alle kommerziellen betrieblichen Aufgaben,
- Textverarbeitungssystem,
- wiss.-techn. Rechner (Dialog-Arbeitsplatz),
- universelles Terminal für Dialogarbeit und/oder Stapelübertragungen,
- Lehrmittel für Hoch- und Fachschulen,
- Arbeitsplatz für NC-Programmierer,
- Konvertierung von Daten zwischen Diskette/Kassette und ½"-Magnetband (in beide Richtungen).

Für diese Anwendungen erfolgt die Massenproduktion der Geräte bzw. die Pflege und Erweiterung von SIOS.

Darüberhinaus schafft die hochleistungsfähige Hardware der Bürocomputer technische Voraussetzungen, mit denen Aufgabengebiete erschlossen werden können, für die SIOS 1526 nicht gedacht/zugeschnitten ist.

Das sind:

- Anschluß "anwendereigener" Geräte, Meßstellen o.ä., für die spezifische driver vom Anwender zu schreiben sind, wobei solche Systeme mit höheren Sprachen programmierbar sein sollen.

  SIOS gestattet derartige driver nicht, es ist kein "offenes" Betriebssystem.

- Nutzung als (Mikrorechner-) Entwicklungssystem für den Prozessor U 880/das System K 1520.

  SIOS bietet hierzu eine relativ breite Unterstützung (CASS, SASS, BKTR zu MRES 20), es fehlen jedoch für diese Zwecke geeignete höhere Programmiersprachen.
  Der Bedarf an geeigneten Mikrorechner-Entwicklungssystemen für K 1520 ist sehr hoch, durch MRES 20 nicht zu decken.

- Nutzung als Mikrorechner-Entwicklungssystem für

  . Ein-Chip Rechner U 881/882

  . SYSTEM U 8000

Diese Aufgaben werden durch SIOS nicht unterstützt.
Um diese Aufgabengebiete rationell zu erschließen, wurde UDOS 1526 als zweites Betriebssystem (neben SIOS 1526) für die Bürocomputer entwickelt.

Beide Systeme existieren parallel, wobei SIOS physisch immer mit den Geräten geliefert wird, da der technische Kundendienst auf SIOS basiert!

Zweck von UDOS 1526 ist es demnach, Bürocomputer A 5120/30 so zu "komplettieren", daß diese arbeiten können als:

- Steuerrechner für kundeneigene Systeme, wobei der Anwender die driver "seiner" angeschlossenen Geräte/Baugruppen selbst schreiben und testen muß;

- Mikrorechnerentwicklungssysteme für die Prozessoren U 880, U 881/882 und Z 8000 bzw. die abgeleiteten Systeme sowie deren Rechner.

Aus dieser Zweckbestimmung, der parallelen Existenz von SIOS (mit abweichenden Datenformaten, Programmstrukturen u.ä.) und einem weiteren, ab Mitte 1984 zu erwartenden Betriebssystem, das u.a. den Anschluß zu den "Nachfolgern der Bürocomputer" herstellt sowie der Tatsache, daß die technischen Mittel der Bürocomputer ständig weiterentwickelt werden, ergeben sich eine Reihe von Problemen, wie z. B.:

- Datenkonvertierung auf Diskette zwischen SIOS und UDOS, 1984 einem dritten Betriebssystem,

- Programmtransformation von Programmen für U 880 von/nach MRES 20 einschließlich geeignetem Datenträger,

- Besprechen von P-ROM für alle Systeme/Prozessortypen,

- Fehlen ausreichender Schulungsmöglichkeiten für UDOS, nicht ausreichende Kapazitäten für Beratung/Projektierungsleistungen bei den Robotron-Vertriebsbetrieben.

Der Entwickler und Verkäufer, VEB Robotron-Buchungsmaschinenwerk Karl-Marx-Stadt, bemüht sich, diese - und weitere Probleme - erfolgreich zu lösen.

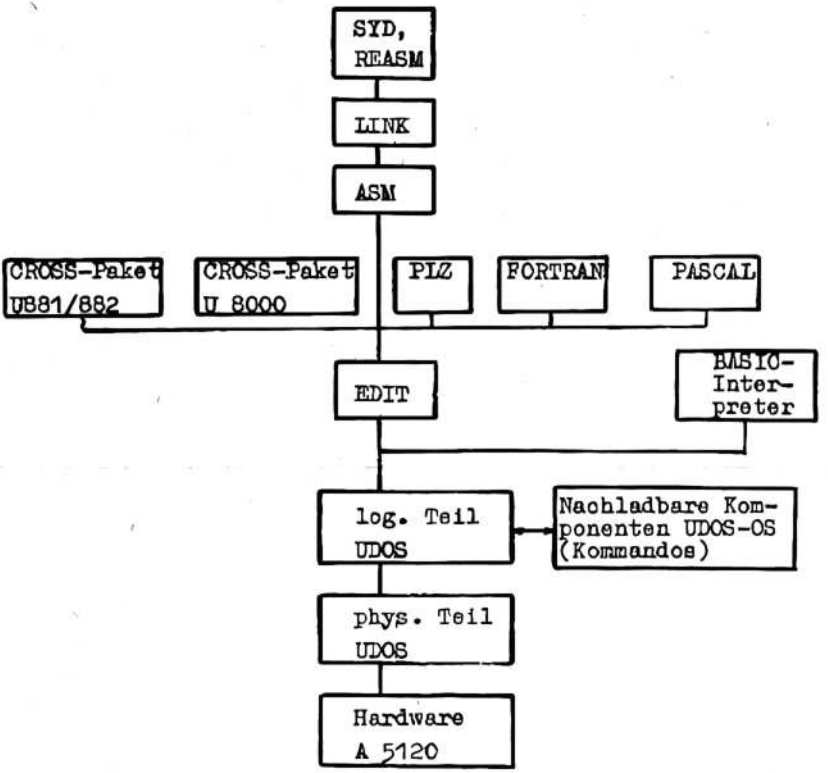
1.2. Programmiergerät PRG 700 mit UDOS 1526 (siehe auch Anlage 1)

Das Bildschirmprogrammiergerät PRG 700 ist ein Erzeugnis des VEB Numerik "Karl-Marx". Es wird als Programmier- und Testeinrichtung in der Steuerungstechnik eingesetzt. Das PRG 700 - eigene, diskettenorientierte Betriebssystem BS 600 gestattet es, Anwenderprogramme für speicherprogrammierbare Steuerungen (PC 600, PC 610, PC 700), für CNC-Steuerungen (CNC 600, CNC-H) und für Industrieroboter-steuerungen (IRS 700 K) zu erstellen, zu testen, zu archivieren und zu dokumentieren. Der Nutzer des PRG 700 ist darüberhinaus in der Lage, anstelle des BS 600 das Betriebssystem UDOS 1526 über eine Systemdiskette zu laden und mit dessen Hilfe das Gerät multivalent zu nutzen. Prinzipielle Unterschiede in der UDOS-Handhabung zwischen Bürocomputer und PRG 700 bestehen nicht, so daß vorliegendes Systemhandbuch für beide Einsatzfälle gültig ist. Einige Besonderheiten, die sich aus einer geringfügig geänderten Tastaturbelegung, einem EPROM-residenten Debugger und einer Einschaltdiagnose des PRG 700 ergeben, sind in Anlage 1 beschrieben.

Der Vertrieb des PRG 700 mit UDOS und die Anwenderbetreuung werden vom VEB Numerik Karl-Marx-Stadt realisiert. Auskünfte zur Software erteilt:
Forschungszentrum des Werkzeugmaschinenbaus
9048 Karl-Marx-Stadt
Annaberger Straße 231
Tel.: 599 328

1.3. Struktur des Systems

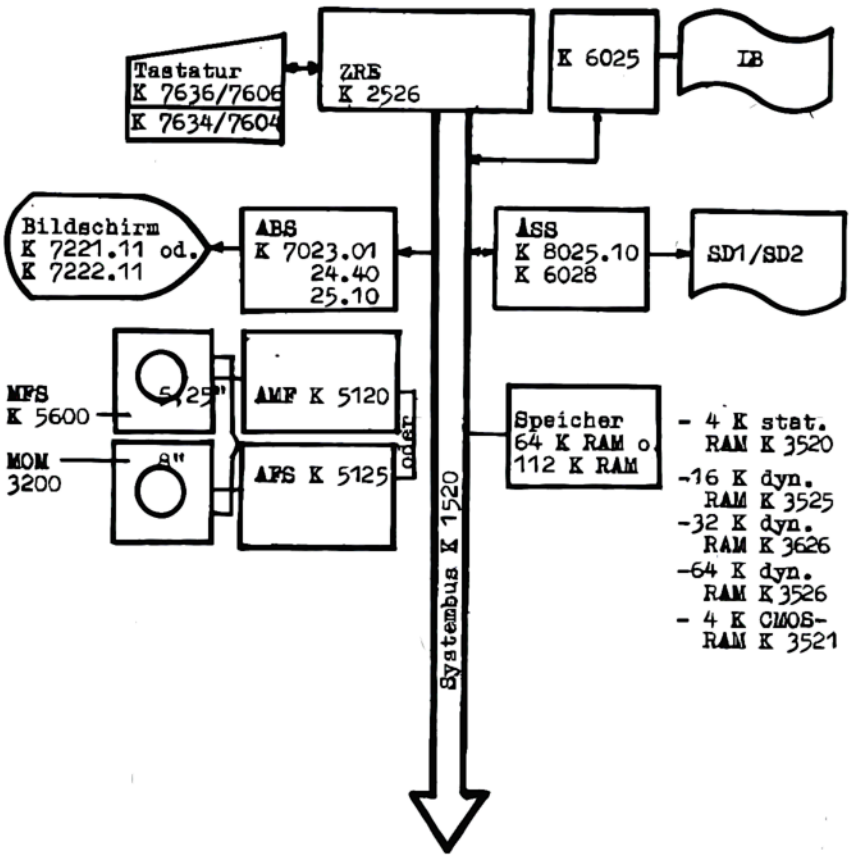
Das System stellt sich dem Anwender wie folgt dar:



2. Hardware

2.1. Übersicht

UDOS 1526 ist anwendbar auf Bürocomputer robotron A 5120/30 bzw. abgeleiteten Terminals, die folgende Komponenten besitzen:



2.2. Bemerkungen zu den Komponenten

- Speicherausstattung

  Mit UDOS ist es möglich, einen Speicher von 64 K Byte zu adressieren, d. h., zu nutzen.

  Zur Arbeit wird eine **Mindest**ausstattung mit 32 K gefordert.
  Geräte mit einer Ausstattung von > 64 K RAM können benutzt werden; der Speicher > 64 K bleibt unbenutzt.

- Bildschirm

  Alle Typen können genutzt werden.

- Drucker

  Außer dem Drucker robotron 1152 können auch die Drucker robotron 1157, beide mit PIO- oder IFSS-Anschluß, genutzt werden.

- Diskettenlaufwerke

  UDOS unterstützt die Diskettenformate 5,25" und 8".
  Genutzt werden 26 Sektoren/Spur (softsektorierte Diskette).

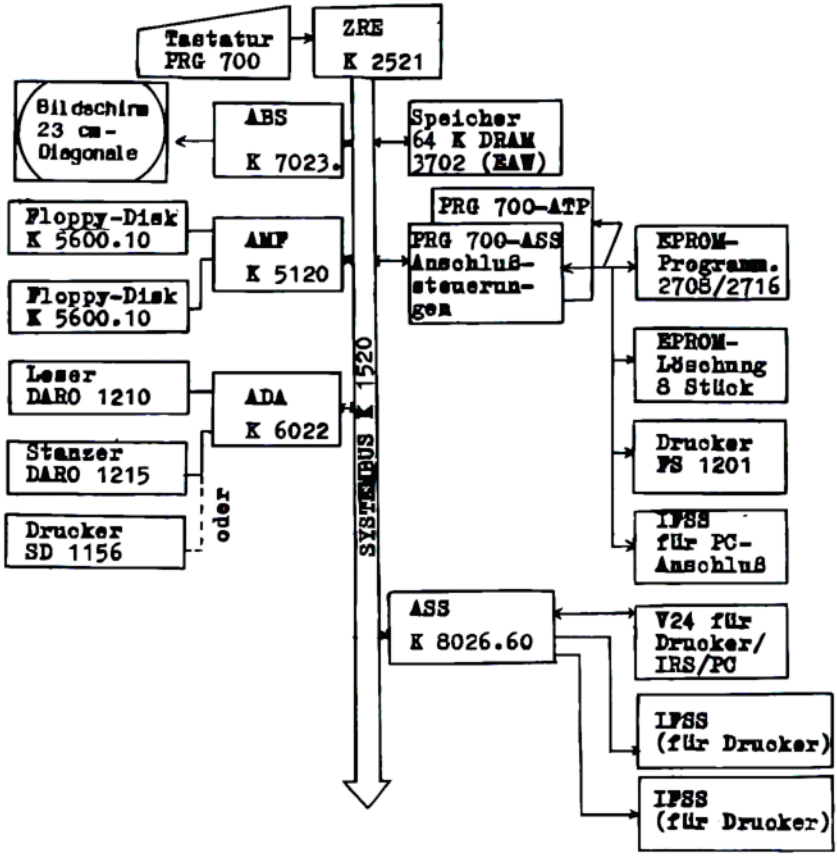
- Tastatur

  Das UDOS-System ermöglicht eine Arbeit mit den Tastaturen K 7606, K 7636 (K 7604 und K 7634; aber Tastenbeschriftung beachten) (siehe Pkt.7).

- Andere Baugruppen

  Andere Baugruppen (z.B. MB-Kassette, 1/2"-MB u.a.) können physisch am Gerät angebracht sein. Sie werden z.Z. durch das System noch nicht unterstützt.
  An Treibern für MB-Kassette und P-ROM-Programmiergerät wird gearbeitet.

2.3. Übersicht PRG 700



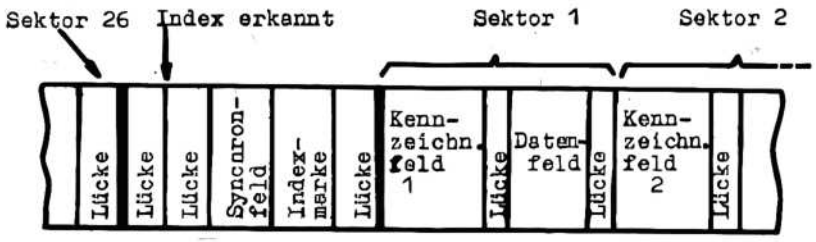
3. Diskettenaufbau und -organisation

Die zum Einsatz kommenden Disketten 5,25" und 8" (Spur- und Aufzeichnungsdichte beachten!) entsprechen in ihrem physischen Aufbau den in Punkt 1 bis 4 der KROS-Standards (KROS 5108 bzw. KROS 5110) beschriebenen Form.

Abweichend dazu werden jedoch an jeden Datensatz 6 Byte angehängt, die einen Zeiger zum vorhergehenden Satz (Back-Pointer) und einen Zeiger zum folgenden Satz (Fore-Pointer) sowie ein CRC-Zeichen enthalten. Dadurch sind die Sätze einer Datei logisch miteinander verkettet und gestreut auf der Diskette untergebracht. Die Verwaltung der Zeiger erfolgt durch das Betriebssystem UDOS selbständig.

Die Aufzeichnung der Daten erfolgt im ASCII-Code.
Die Disketten sind fest sektoriert, wobei die Disketten 40 bzw. 77 Spuren mit je 26 Sektoren zu 128 Bytes enthalten.
Es können mehrere Sektoren zu einem Datenblock (Satz) zusammengefaßt werden.
Ein Satz besteht aus 1, 2, 4, 8 oder 16 Sektoren und hat damit eine Länge von 128, 256, 512, 1024 oder 2048 Bytes.
Durch Zusammenfassen mehrerer Sektoren wird eine höhere **Zugriffseffektivität erreicht.**

Physischer Spuraufbau Diskette



Aufbau Sektoren 1 - 26

Kennzeichnungsfeld analog KROS

Bezeichnung	Byteanzahl 8"	5,25"
Synchronfeld	6	12
-	-	3
AM1	1	1
Spurnummer	1	1
-	1	-
Sektornummer	1	1
physische Satzlänge	1	1
CRC	2	2

Datenfeld

Bezeichnung	Byteanzahl 8 "	5,25"
Synchronfeld	6	12
-		3
AM 2	1	1
physischer Satz	128	128
CRC	2	2
Fore-Pointer	2	2
Back-Pointer	2	2
CRC	2	2

4. Der ASCII-CODE

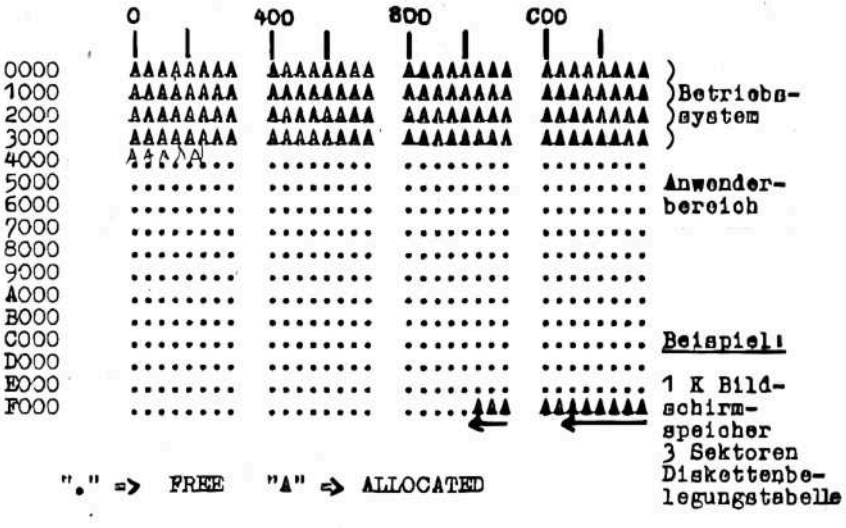
Zeile \ Spalte	0	1	2	3	4	5	6	7
00	NUL	DLE	SP	0	@	P	`	p
01	SOH	DC1	!	1	A	Q	a	q
02	STX	DC2	"	2	B	R	b	r
03	ETX	DC3	#	3	C	S	c	s
04	EDT	DC4	$	4	D	T	d	t
05	ENG	NAK	%	5	E	U	e	u
06	ACK	SYN	&	6	F	V	f	v
07	BEL	ETB	'	7	G	W	g	w
08	BS	CAN	(	8	H	X	h	x
09	HT	EM	)	9	I	Y	i	y
0A	LF	SUB	*	:	J	Z	j	z
0B	VT	ESC	+	;	K	[	k	{
0C	FF	FS	,	<	L	\	l	|
0D	CR	GS	-	=	M	]	m	}
0E	SO	RS	.	>	N	^	n	~
0F	SI	US	/	?	O	_	o	DEL

5. Grundaufbau des Betriebssystems

Entsprechend der Gerätekonfiguration mit Folienspeichern gliedert sich das System zunächst in den residenten Teil, der nach Einschalten des Gerätes durch einen Ladevorgang in den Hauptspeicher gebracht wird und dann ständig zur Verfügung steht. Ein weiterer Teil, der temporäre Teil des Systems, befindet sich auf der Diskette und wird je nach Bedarf geladen und gestartet. Dieser Teil umfaßt alle Dienst- und Hilfsprogramme, alle zusätzlichen Gerätetreiber, alle Programmaufbereitungsprogramme (Sprachen) oder alle vom Anwender erstellten Phasenmoduln.

HS-Aufteilung:

MEMORY ALLOCATION MAP



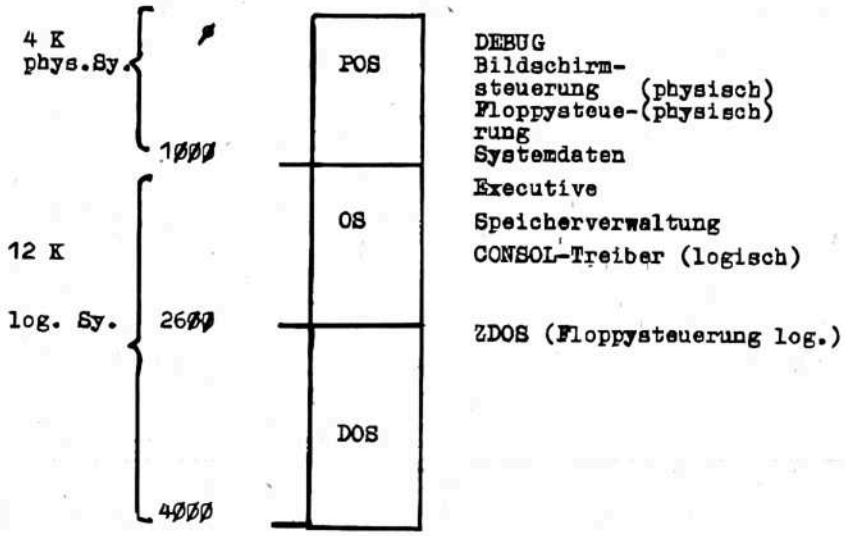
0000 - 3FFF Betriebssystem (Kern)
4000 - .... Anwenderbereich
FFFF 1 K oder 2 K Bildschirmspeicher
..... 3 Sektoren Diskettenbelegungstabelle je angeschlossenem Laufwerk

Residenter Teil des Betriebssystems:

Diesen Bereich muß man nochmals in zwei Teile, den physischen und den logischen, untergliedern. Beim Einschalten des Gerätes oder der Erzeugung eines "RESET"-Impulses an der ZRE des Bürocomputers wird zunächst der physische Teil von der

Diskette in den Hauptspeicher geladen. Dieser Teil enthält die Hardwaresteuerung für Tastatur, Bildschirm und Floppysystem sowie den Debugger (Monitor), der sich nach der Geräteinitialisierung mit dem Prompt-Zeichen "+" meldet.

Speicheraufteilung



6. Physisches Betriebssystem

6.1. Debuggerkommandos

In diesem Programm stehen Kommandos für die Maschinenprogrammtestung, Speicher- und Registeranzeige bzw. Änderung, Portein- und -ausgabe, sowie physische Diskettenarbeit zur Verfügung.

Kommandobeschreibung:

Q Bewirkt das Verlassen des Debuggers in das übergeordnete Programm (ist nach Einschalten noch nicht möglich).

Der Aufruf kann aus dem logischen Teil (OS) erfolgt sein. Er ist aber auch von jedem anderen Programm aus über die Ansprungadresse ØBFA möglich. Dazu muß jedoch vorher noch die Rückkehrsdresse in den Speicherzellen ØFB5/ØFB6 eingetragen und Bit 5 der Speicherzelle ØFC4 gesetzt werden.

O startet den Lade- und Initialisierungsvorgang des logischen Betriebssystemkerns und geht in dieses über. Danach meldet es sich mit dem Promtzeichen "%".

R "Registername"

zeigt entweder alle internen Register an oder nur das ausgewählte und bietet dann die Möglichkeit der Änderung. Wenn keine Änderung erwünscht ist, muß nur "ET1" betätigt werden. In beiden Fällen wird jedoch auf das nächste Register weitergeschaltet. Der Rücksprung ist durch "Q" möglich.

D ("Adresse" ["Länge"])

Anzeige des Inhaltes einer Speicherzelle und Möglichkeit der Neueingabe sowie nachfolgender Weiterschaltung auf die nächste Adresse, Soll keine Änderung erfolgen, muß nur "ET1" betätigt werden. Durch Eingabe von "^" wird die vorherige angezeigt und durch "Q" die Speicherdarstellung verlassen.

Bei Angabe der Länge wird eine hexadezimale bzw. textmäßige Darstellung eines entsprechenden Bereiches vorgenommen. Das "Roll-on" der Ausgabe kann dabei durch Betätigung einer beliebigen Taste gestoppt und wieder fortgesetzt oder durch die Taste "ET2" abgebrochen werden.

M ("Zieladresse" "Quelladresse" "Länge")

speichert einen Bereich mit entsprechenden Parametern um.

PO ("Portadresse" "Bitmuster")

Führt die Ausgabe des angegebenen Bitmusters zum entprechenden Port durch.

PI ("Portadresse")

Abtragen eines Portes und Möglichkeit der Ausgabe

B ["Adresse"]

Prüfpunktregister für Programmtestung mit entsprechender Adresse laden. Keine Adresse angegeben, so wird das Prüfpunktregister gelöscht.

J ["Adresse"]

Sprung zu angegebener bzw. wenn nicht angegeben, zu der im PC-Register eingetragenen Adresse. Im Stack wird vorher die Rückkehradresse eingetragen, so das die Rückkehr durch einen "RET"-Befehl möglich ist.

Es wird der Systemstatus eingestellt, d. h.:

SP = ØCFE, I-Reg. = ØF, Interruptmode = 2,
Interrupt enable

Beim J-Kommando wird ein eingestellter Breakpoint nicht beachtet. Dieses Kommando sollte nur zum Abarbeiten bereits getesteter (Unter)programme, nicht aber zum Austesten von Programmen angewendet werden (siehe auch G-Kommando).

G ["Adresse"]

Dieses Kommando dient dem Austesten von Programmen. Bei Erreichen eines Breakpointes wird die Programmabarbeitung unterbrochen.
Die Reaktion ist analog dem J-Kommando, jedoch wird nicht der Systemstatus, sondern der in den Registern abgespeicherte eingestellt.
Die Rückkehr ist durch den Befehl "RST 38" möglich.

I ["D"]

Ermöglicht oder verbietet bei angegebenem Parameter die Unterbrechungen bei Programmausführung.

L ("Diskadresse", "Hauptspeicheradresse", "Länge")

Lesen eines Records (Satzes) von Diskette.

S ("Diskadresse", "Hauptspeicheradresse", "Länge")

Schreiben eines Records (Satzes) auf Diskette mit einer Länge von 1 ... ØDØØ Byte (26 Sektoren)

Die Diskettenadresse wird als "Word" angegeben (hexadezimal)

H-Byte: Spurnummer Ø ... 27 (bzw. 4C bei 8"-Diskette)

L-Byte: Bit Ø ... 4 Sektornummer Ø ... 19

Bit 5 ... 7 Laufwerksnummer Ø ... 3

Zu beachten: Die Sektornumerierung beginnt in der Spur bei Ø und endet bei 25!
Alle Sektoren, die geschrieben werden, erhalten die gleichen Zeigerinformationen.
Also immer nur <u>einen</u> Record (das können mehrere Sektoren sein) mit einem S-Kommando schreiben!

Beispiel:

```
%D
D
+R                Registeranzeige
A  B  C  D  E  F  H  L  I   IX   IY   PC   SP
00 00 00 00 00 00 00 00 00 0000 0000 0000 0000
00 00 00 00 00 00 00 00
D FCA             ;ANZEIGE DER LAUFWERKSBELEGUNG
0FCA 16  Q        ;0=8"; 1,2=5"; 3=NICHT VORHANDEN
+L 1550 7000 300;Lädt von Laufwerk 2 Spur 15 ab Sektor
                  1Ø (d. h. dezimal: auf der 21. Spur ab
                  17. Sektor)
                 ;300 Byte (6 Sektoren) in den Haupt-
                  speicher
                 ;ab Adresse 7ØØØ in den Arbeitsspeicher

+Q
%
```

6.2. Systemdaten

Im Hauptspeicherbereich von ØCØØ ... ØFFF befinden sich die Systemdaten. Einige Adressen mit allgemeiner Bedeutung sollen hier angegeben werden:

Name	Adresse	
Stack	ØCØØ-ØCFFH	Debuggerstack
OUTBU	ØD04-ØD83H	Datenpuffer für BS-Ausgabe
INBU	ØD88-ØE87H	Datenpuffer für Tastatureingabe
PTR	ØE9F-ØEA2H	Rückwärts- und Vorwärtspointer für Diskettenrecord
IL	ØFØØ-ØF8ØH	Interrupttabelle für Anwenderhardware
INBUAD	ØFBB-ØFBC	Pointer für Ende der Eingabe im Puffer
PROMPT	ØFC1	Promptzeichen für Kommandoeingabe
CHRDEL	ØFC3	Code für Zeichenlöschung (normal ØBH)
DEBFL	ØFC4	Debuggerflag (siehe Q-Kommando)
DISKCO	ØFCA	Disketten-Konfiguration für jedes Laufwerk 2 Bit Dabei Ø: nicht vorh. 1: MFM 40 Sp (MFS) 2: FM 77 Sp (8"-LW)
BSA	ØFD7	High-Teil d. Bildschirmanfangsadresse
MADR	ØFDE-ØFDF	Markenadresse für Bildschirmkursor
RSTRET	ØEEC-ØEED	Rückkehradresse für RST-38H
NMIRET	ØEEE-ØEEF	" für NMI
CTCIL	ØFEØ-ØFEF	Interruptliste für System CTC (Kanal 1 noch frei)

Belegte E/A - Adressen im Bürocomputer A 5120/A 5130 mit UDOS

Baugruppe	E/A - Adresse Bit 7...4	3...Ø	Bemerkung
ZRE	Ø	Ø...3	CST 1 Tastatur
		4...7	CST 2 Tastatur
		8...B	BS - PIO
		C...F	CTC, Kanal Ø ØC 1 ØD frei 2 ØE 3 ØF
FD K 5120	1	Ø...B	
Monitor	4	Ø...3	
Drucker K 6028	5	Ø...3	SIO
		4...7	PIO
		8...B	CTC
		C...F	frei
Lochband K 6025	9	Ø...7	

Bei Anschluß weiterer Hardware sollten vorzugsweise folgende Adressen belegt werden, um die Kompatibilität zu SIOS zu sichern:

KMBG	3	Ø...F	
Sif 1000-StE (K 6022)	7	Ø...3	z.B. für 1156
EPROM-Progr.	E	Ø...7	

Stand: März 1984

6.3. Physische Programmodule

Für die universelle Anwendung stehen über einen Sprungverteiler einige Unterprogrammroutinen zur Verfügung.

Name	Adresse	Benutzung
INA	OBE8H	Test, ob ein Zeichen von Tastatur eingegeben wurde (Dann "NZ") und Übergabe im A-Register
OUTA	OBEBH	Ausgabe eines im A-Register übergebenen Zeichens
WRTXT	OBF1H	Ausgabe eines Textes bis 256 Zeichen Dabei: (HL) = Länge (Abbruch bei "CR") HL+1= Anfangsadresse (Zerstörung A,B-Register)
LDNECH	OBF4H	Suchen und Laden (ins A-Register) des nächsten im Eingabepuffer vorliegenden Zeichens (Zerstörung HL-Register)
COMIN	OBF7H	Ausgabe des Prompt-Zeichens und warten auf die Eingabe einer Zeichenkette, die mit ET1 ("CR") abgeschlossen wurde und ablegen im Eingabepuffer (ØD8AH-ØE89H). Das erste Zeichen wird im A-Register übergeben. (Zerstörung A,B,HL-Register)
DEBUG	OBFAH	Ansprungsadresse für Debugger (siehe Q-Komm.)
Floppy	OBFDH	Physischer Floppytreiber-Parametorübergabe erfolgt über Vector (IY-Vectoranfangsadresse) (IY) - wird nicht benutzt (IY+1) - Requestcode: OAH-Lesen, OBH-Schreiben IY+2),(IY+3) - Datenbereichsanfang (IY+4),(IY+5) - Datenbereichslänge (Ø...DØØH) (IY+6),(IY+7) - Rückkehradresse (IY+8),(IY+9) - Errorreturn-Adresse (IY+10) - RETURN-Code (IO:8ØH, bei Fehler: C1...C6) (IY+11),(IY+12) - Diskadresse (sh. S-Kommando) Alle Register außer A'...F' werden zerstört.

Die Arbeit mit dem physischen Floppytreiber:

Durch die universelle Vektorschnittstelle ist die Arbeit mit jedem Treiber gleich. Zunächst soll anhand dieses Treibers die grundsätzliche Arbeitsweise verdeutlicht werden. An anderer Stelle wird dann der Aufbau eigener Gerätetreiber und die Einbindung in das Betriebssystem beschrieben.

Vor dem Unterprogrammaufruf des Treibers muß zunächst ein Datenbereich (Vektor) aufgebaut werden, der für den Parameteraustausch genutzt wird. Im IY-Register wird dann die Anfangsadresse übergeben. Die einzelnen Bytes haben folgende Bedeutung:

Requestcode:

- Stellt dar, "was" gemacht werden soll.
  Ist Bit Ø = Ø, so gibt der Treiber die Steuerung erst nach vollständiger Abarbeitung der Anforderung zurück. Ist aber Bit Ø = 1 (z.B. Lesen ØA → ØB), so führt der Treiber nach Aufruf nur alle zu diesem Zeitpunkt möglichen Aktionen durch, gibt die Steuerung über einen "RET"-Befehl mit Returncode = Ø zurück und arbeitet dann durch Unterbrechungen zeitparallel bis zur vollständigen Aufgabenbearbeitung weiter.

Completion-Return

- Stellt eine Adresse dar, die angesprungen wird, wenn der Treiber den Request vollständig abgearbeitet hat.
  Im gegebenen Fall muß Registerrettung beachtet werden.

Error-Return

- Stellt analog der Completion-Returnadresse die Programmfortsetzung für bei der Abarbeitung aufgetretene Fehler dar.

Returncode

- Signalisiert das Ergebnis der Arbeit

ØØ - noch nicht fertig	C3 Schreibschutz
8Ø - Operation erfolgreich	C4 Sektoradressfehler
C1 - Falsche Anforderung	C5 Spurfehler
C2 - Laufwerk nicht bereit	C6 CRC-Prüfsummenfehler

Die Parameter in (IY+11) und (IY+12) werden bei physischen Treibern spezifisch genutzt und bei der Floppyarbeit für die Diskettenadresse verwendet.

7. UDOS-Executive und UDOS-Kommandos

7.1. UDOS-Executive (OS)

Die Executive, im folgenden immer mit OS bezeichnet, gehört neben dem zentralen Diskettentreiber (ZDOS) zum logischen Teil von UDOS.
Die logische Ebene ist die Standardarbeitsebene des Nutzers von UDOS; das Arbeiten direkt auf physischem Niveau sollte dem Havariefall vorbehalten bleiben.

7.1.1. Ladevorgang und Initialisierung des Systems

Nach Einschalten der Maschine bzw. Erzeugung eines RESET-Signals wird der Lade-ROM der ZRE K 2526 aktiviert. Das Programm des Lade-ROMs ist so ausgelegt, daß zunächst - beginnend beim Diskettenlaufwerk Ø - eine Systemdiskette gesucht wird.
Von dort wird der Systemlader geladen und angesprungen. Gleichzeitig schaltet sich der Lade-ROM ab. Der Systemlader seinerseits lädt den physischen Teil von UDOS (die gesamte Spur 2) und startet diesen Teil. UDOS meldet sich mit dem physischen Systemprompt '+'.
Diese Ebene ist gleichzeitig die Kommandoebene für den Systemdebugger (siehe auch 6.1.).
Durch das Kommando 'O' (Organize) wird der UDOS-Bootstrap-Lader aktiviert, welcher die beiden Dateien OS und ZDOS in den Arbeitsspeicher lädt und zum Eintrittspunkt des OS springt.
Jetzt beginnt die Initialisierung des logischen Teils von UDOS. Dazu gehören eine ganze Reihe von Maßnahmen, wie z.B.

- Austesten des Anwenderspeichers und Belegung der Bit-Map des Speicherverwalters (MEMMGR), wodurch Betriebssystembereiche ab sofort als belegt gelten.
- Initialisierung der Systemkonsole, d.h. Voreinstellung von konsolspezifischen Steuerzeichen, wie z.B.

```
LINDEL    line delete      (7FH)
CHRDEL    character delete (ØBH)
```

  oder von Anpassungszeichen, wie z.B.

```
NULLCT    number of null count(1)
LFCNT     line feed count      (1),
```

  die nach jedem CR (ØDH) eingefügt werden.
  Ebenso gehört dazu die automatische LF-Einfügung und der ECHO-Modus

```
AUTOLF ON
ECHO   ON.
```

Alle diese Werte bzw. Betriebsmodi sind mit dem SET-Kommando änderbar.

- Initialisierung des primären Dateisystems, d.h. Festlegung von ZDOS als "Masterdevice".
  Damit werden alle unvollständig qualifizierten Dateibezeichnungen (wo kein Treiber explizit angegeben ist) mit ZDOS verbunden und die zugehörigen E/A-Requests zu ZDOS gelenkt.
- Abarbeiten einer Kommandodatei mit dem Namen OS.INIT. Diese Datei ist dem Anwender über den Editor zugänglich. Er kann somit eine benutzerorientierte Systeminitialisierung durchführen.

Standardmäßig hat OS.INIT folgenden Inhalt:

```
I
DATE 840102
ECHO
X *  **** BETRIEBSSYSTEM UDOS ****
X *           VERSION 3.0
X *
X *
X * DATUM EINGEBEN!
V
%
```

Für den Systemanlauf ist damit erforderlich, daß auf einer Systemdiskette neben OS und ZDOS auch noch die Dateien DO, OS.INIT sowie die in OS.INIT enthaltenen externen Kommandos (hier z.B. DATE und ECHO) vorhanden sind.

- Den Abschluß der Initialisierung bildet die Systemausschrift
  UDOS BC 5120
  und die Ausgabe des Systemprompts '%'.

7.1.2. Speicherverwalter (MEMMGR)

Jedes Programm, das in UDOS abgearbeitet werden soll, benötigt einen bestimmten Speicherbereich. Dieser wird beim Linken fixiert. Die Segmentgrenzen sind im Datei-DESCRIPTOR vermerkt.
Wenn das Programm geladen werden soll, wird geprüft, ob im Speicher der erforderliche Adreßbereich überhaupt frei ist. Wenn das der Fall ist, wird das Programm geladen und der Speicherbereich als belegt markiert. Wenn der Speicherbereich bereits belegt ist, dann wird ein "Überladen" verhindert und die Speicherschutzverletzung mit der Meldung

MEMORY PROTECT VIOLATION

angezeigt.

Alle diese Aufgaben werden im Speicherverwalter (memory manager) des OS umgesetzt. Insbesondere werden dort die beiden internen Kommandos

ALLOCATE und DEALLOCATE realisiert.

Der Speicherverwalter ist ein selbständiges Unterprogramm des OS und kann über einen CALL zur Adresse 1009H angesprungen werden.

Im A-Register wird das Kommando übergeben (A=Ø ---> Allocate, A=1 ---> Deallocate). In den Registerpaaren bedeutet

HL	untere Adreßgrenze
DE	obere Adreßgrenze
BC	Blockgröße.

Bei Deallocate entfällt DE und HL bedeutet Anfangsadresse des Speicherblockes.
Nach Ausführung der Kommandos wird in A zurückgemeldet, ob die Operation erfolgreich war:

A = 8ØH	okay
A ≠ 8ØH	Block nicht vorhanden bei Allocate bzw. nicht alle Blöcke waren belegt (bei Deallocate).

Der Speicherverwalter arbeitet mit einer Bit-Map zusammen, in der jedes Bit 128 Byte des Arbeitsspeichers repräsentiert. Wenn ein Bit gesetzt ist, ist der korrespondierende Bereich von 128 Byte belegt, ist das Bit gelöscht, dann ist er frei.

Die Bit-Map kann mit dem Kommando "DISPLAY" auf Konsole angezeigt werden.
Nichtexistierender Speicher (wenn z.B. eine 16K Byte RAM-Steckeinheit entfernt wurde) wird in der Bit-Map automatisch als belegt gekennzeichnet.

7.1.3. E/A-Struktur

Die E/A-Struktur des Betriebssystems UDOS bietet eine hohe Flexibilität. Insbesonder das OS unterstützt durch seine standardisierte E/A-Schnittstelle (I/O-Vector) die einfache Implementation zusätzlicher Gerätetreiber.

Alle E/A-Requests werden als CALL zum Systemeinschnittspunkt (1ØØ3H) umgesetzt. Die Parameter sind in Form eines Vectors, dessen Anfangsadresse im Register IY stehen muß, zu übergeben. Der Vector besitzt folgendes Format:

	Byte	Inhalt
IY --->	1	Nummer des logischen Gerätes
	2	Requestcode
	3,4	Datenbereichsanfang
	5,6	Datenbereichlänge
	7,8	Rückkehradresse
	9,10	Fehlerrückkehradresse
	11	Fertigstellungscode
	12,13	Zusatzparameterinformationen (optional)

E/A-Requests beziehen sich damit immer auf ein logisches Gerät. Das OS prüft beim Systemcall (1ØØ3H) den Request-Vector und übergibt die Steuerung an den im Byte Ø des Vectors spezifizierten logischen Treiber.

OS ist in der Lage, mit max. 20 logischen Geräten zu arbeiten. Jedes logische Gerät (=Treiber) muß, bevor es angesprochen werden kann, aktiv sein und per Definition eine logische Nummer zugeordnet bekommen.

OS-intern sind diese Informationen in einer Tabelle (aktive device table = ADT) gespeichert. Sie enthält für jeden Treiber eine Eintragung (Treibername, Entry-Point, logische Nummer).

Die Eintragung (=Aktivierung) eines Treibers in diese Tabelle erfolgt mit dem Kommando ACTIVATE, die Austragung (=Deaktivierung) mit dem Kommando DEACTIVATE. Die gesamte Tabelle ist mit dem Kommando LADT auf Konsole anzeigbar.

Einem aktiven Gerätetreiber kann mit dem DEFINE-Kommando bzw. mit einem ASSIGN-Request eine logische Nummer (UNIT) zugeordnet werden. Bei der Initialisierung des Systems erfolgt für die Standardgeräte bereits eine Voreinstellung:

Treiber	UNIT
ZDOS	Ø 4 5 6 7 ... 19
CON	1 2 3
NULL	2Ø

UNIT Ø ist für Systemfunktionen reserviert und im eigentlichen Sinne kein Gerätetreiber. UNIT 1, 2 und 3 werden als CONIN, COUNUT und SYSLST bzeichnet und realisieren die Konsoleingabe (Tastatur), Konsolausgabe (Bildschirm) und das Listgerät (Bildschirm oder Drucker).
UNIT's 4 bis 20 können für Anwenderprogramme verwendet werden.

In den Gerätetreibern sind die E/A-Requests in physische Aktionen der betreffenden Geräte einzusetzen. Solche Requests sind z.B.:

INITIALIZE	Gerät in Grundzustand bringen
ASSIGN	Zuweisung einer logischen Nummer
OPEN	Datei eröffnen
CLOSE	Datei schließen
WRITE BINARY	einen Datensatz schreiben
READ BINARY	einen Datensatz lesen
usw.	

Der größte Umfang von Requests ist im ZDOS realisiert (sh. auch 8.2.6.). In anderen Gerätetreibern ist es im allgemeinen erforderlich, nur eine Untermenge dieser Requests von ZDOS zu implementieren, vor allem dann, wenn dort keine Dateiverwaltung erfolgt (wie z.B. Drucker, LB).

Bei der Programmierung von E/A-Operationen in Anwenderprogrammen muß bekannt sein, welche Requests in den angesprochenen E/A-Treibern umgesetzt werden können. Nur diese dürfen im Anwenderprogramm vorkommen, da das OS entsprechend der logischen Nummer im Byte Ø des E/A-Requests diesen sofort zur Realisierung an den Treiber übergibt. Wird ein Request programmiert, der im betreffenden Treiber überhaupt nicht implementiert ist, dann tritt ein E/A-Fehler (ERROR C1) auf.

Das OS bietet die Möglichkeit, von einem Programm bzw. Kommando aus ein beliebiges anderes Programm aufzurufen und abzuarbeiten.

Dazu bedient es sich der gleichen E/A-Schnittstelle wie bei E/A-Operationen. Der Aufruf erfolgt mit einem CALL zum System (1ØØ3H), wobei der Request-Vector in diesem Fall ohne Zusatzparameterbyte benutzt wird.
Der Vector hat nun folgendes Format:

	Byte	Inhalt
IY--➔	1	Ø Systemrequest (keine E/A!)
	2	- nicht benutzt
	3,4	Adresse der Kommandokette
	5...8	- nicht benutzt
	9,10	Fehlerrückkehr-Adresse
	11	Fertigstellungscode

Um ein Programm aufzurufen, ist eine Kommandokette aufzubauen in der gleichen Art, als ob sie über Konsole eingegeben wurde, im Speicher zu vereinbaren. Die Anfangsadresse dieser Kette ist im Byte 2,3 des Request-Vectors zu vereinbaren und die Adresse des Request-Vectors in IY zu übergeben.

7.1.4. Kommandointerpreter

Wenn das OS das Systemprompt '%' ausgibt, wird ein Kommando bzw. eine ganze Kommandokette von der Konsole erwartet. Diese Kommandokette darf maximal 256 Zeichen lang sein einschließlich Endezeichen CR (ØDH). Die Zeichen gelangen in den Eingabepuffer INBU.

Werden mehrere Kommandos eingegeben, so sind sie durch Semikolon';' zu trennen. Nach Eingabe von CR wird die Kommandokette vom Kommandointerpreter abgearbeitet.

Die Arbeit des Kommandointerpreters soll nachfolgend kurz dargestellt werden:

- Es wird vorausgesetzt, daß das Kommando im Eingabepuffer steht.
- Es erfolgt ein Vergleich mit der Liste der internen Kommandos; bei festgestellter Übereinstimmung wird das interne Kommando im OS angesprungen.
- Bei Nichtübereinstimmung wird das Kommando als extern deklariert und als Datei auf dem Master-Device (mittels OPEN-Request) gesucht.
- Die gefundene Datei muß den Typ P (PROCEDURE) besitzen.
- Der Speicherverwalter (MEMMGR) prüft, ob der für die Datei erforderliche Speicherbereich frei ist (Allocation).
- Wenn die Prüfung positiv ausfällt, wird die Datei geladen und dann im Eintrittspunkt gestartet (falls E ≠ Ø und kein Komma hinter dem Kommando).
- Nach Abarbeitung wird der vom Kommandoprogramm belegte Speicherbereich wieder freigegeben (Deallocation). Das Programm selbst bleibt aber im Speicher und auch sein Eintrittspunkt bleibt im OS erhalten. Damit ist dieses Programm ohne erneutes Laden sofort wieder startbar (mehrfach) durch Anwendung des internen Kommandos 'X'.
  Wenn eine Datei nur geladen, aber nicht abgearbeitet wird (Komma hinter Dateiname!), dann bleibt der Speicherbereich belegt. Die Freigabe ist über das Kommando "RELEASE" möglich.

7.2. Beschreibung der UDOS-Kommandos

UDOS verfügt über externe und interne Kommandos. Während sich die internen Kommandos nach dem Initialisieren des Systems ständig abarbeitungsbereit im Speicher befinden, werden die externen vor Abarbeitung in den Speicher geladen.
UDOS-Kommandoeingaben können als Dateien des Typs A (ASCII) auf der Diskette gespeichert werden und mit dem Kommando DO zur Interpretation aufgerufen werden. Externe UDOS-Kommandos sind Dateien vom Typ P (Prozedure).

Nutzerhinweise zur Darstellung:

* Parameter, die wahlweise einmal oder beliebig oft verwendet werden, sind in () runde Klammern, gefolgt von * eingeschlossen

Leerschritt trennt Kommandoname vom Kommandoparameter

; dient als Trennzeichen zwischen den Kommandos (Eingabe mehrerer Kommandos ist möglich, die Zeichenkette darf max. 255 Zeichen sein)

, Komma am Ende des gesamten Kommandos bewirkt nur das Laden des Kommandos im Speicher
Gestartet wird das Kommando mit XEQ.

\+ Parameter,die mindestens einmal verwendet werden müssen, sind in () runde Klammern,gefolgt von + eingeschlossen

[] Optionale Teile der Parameterliste sind durch eckige Klammern eingeschlossen

| (logisches ODER), es darf einer und nur einer der damit verknüpften Parameter verwendet werden

ET1 ist am Ende der Kommandoeingabe zu bedienen

ET2 als wiederholte Tastenbedienung bewirkt den Abbruch Kommandoeingabe bzw. -ausgabe

M (Monitortaste), bewirkt das Aktivieren bzw. Deaktivieren des Druckers

DEL nach einer Kommandoeingabe, bewirkt das Löschen der Eingabezeile

CE	löscht 1 Zeichen der Eingabe
Tastatur	alle Tasten der alpha- und numerischen Tastatur bewirken STOP bei der Kommandoabarbeitung, ein wiederholtes Betätigen einer solchen Taste, startet die weitere Abarbeitung des Kommandos
ESCAPE	entspricht ET2
CARRIAGE RETURN	entspricht ET1
⇧	Umschalttaste für Ziffern und Sonderzeichen
INS MODE	Umschalttaste für Tasten A-Z (Groß- / Kleinbuchstaben)
CI	CONTROL dient der Eingabe von Steuerzeichen (bewirkt Umschalten von Codezeichen in Steuerzeichen)

z.B.: A ⟹ 41 CI, A ⟹01

Tastenfunktionen in UDOS, die auf den Tastaturen K 7636 und K 7634 durch Unterschiedliche Tasten ausgelöst werden:

K 7636	K 7634
ET1	ENTER
ET2	CNCL
CE	⟵
CI	RESET
S1	PF1
.	.
.	.
S9	PF9
M	OFF

'HELP' IST VERFUEGBAR FUER FOLGENDE UDOS-KOMMANDOS:

ACTIVATE		PERIPHERIEGERAET AKTIVIEREN
ALLOCATE	I	SPEICHER RESERVIEREN
BRIEF	I	OHNE KOMMANDOECHO
CAT		FILE-NAME SUCHEN
CLOSE	I	FILE SCHLIESSEN
COMPARE		FILES VERGLEICHEN
COPY		KOPIEREN
COPY.DISK		DISKETTE KOPIEREN
DATE		SYSTEMDATUM SETZEN/ABFRAGEN
DEACTIVATE		PERIPHERIEGERAET DEAKTIVIEREN
DEALLOCATE	I	SPEICHERRESERVIERUNG LOESCHEN
DEBUG	I	AUFRUF DES DEBUGGERS
DEFINE		ZUORDNUNG LOGISCHE EINHEIT-GERAET AENDERN
DELETE		FILE LOESCHEN
DISPLAY		SPEICHERBELEGUNGSPLAN AUSGEBEN
DO		ABARBEITUNG EINES KOMMANDOFILES
DUMP		HEXADEZIMALER AUSDRUCK EINES FILES
ECHO		TEXTAUSGABE
EXTRACT		AUSGABE DER FILE-ATTRIBUTE
FORCE	I	FILE LADEN OHNE BERUECKSICHTIGUNG DER SPEICHER-RESERVIERUNG
FORMAT		DISKETTE FORMATIEREN
HELP		ERLAEUTERUNG DER BENUTZUNG DES HELP-KOMMANDOS
IMAGE		SPEICHERBELEGUNG ALS PROCEDURE-FILE ABLEGEN
INITIALIZE	I	INITIALISIEREN
LADT		AUSGABE DER ACTIVE DEVICE TABLE
MASTER		ANGABE ODER UMDEFINITION DES MASTERGERAETES
MOVE		FILES KOPIEREN
PAUSE		WARTEN AUF TASTENABFRAGE
PRINT		AUSGABE EINER TEXTDATEI
RELEASE	I	SPEICHERRESERVIERUNG LOESCHEN
RENAME		FILE UMBENENNEN
RESTORE_TABS		TABULATOREN AUS FILE UEBERNEHMEN
SAVE_TABS		AKTUELLE TAB'S AUF FILE ABLEGEN
SET		SETZEN VON SYSTEMVARIABLEN UND FILEEIGENSCH.
STATUS		DISKETTENZUSTAND
VERBOSE	I	LANGE BETRIEBSART, MIT KOMMANDOECHO
XEQ	I	START EINES GELADENEN PROGRAMMES
:	I	BERECHNUNG EINES HEXADEZ. ARITHM.AUSDRUCKES

"I" VOR DER KURZBESCHREIBUNG BEDEUTET, DASS DIESES KOMMANDO SYSTEMINTERN IST UND ZUM RESIDENTEN TEIL DES BETRIEBSSYSTEMS GEHOERT.

ASCII	ASCIIZEICHEN -> ASCIICODE
CHAR	ASCIICODE -> ASCIIZEICHEN
ERROR	FEHLERCODE -> FEHLERART

7.2.1. ACTIVATE "GERAETENAME" [ENTRY]

DAS ANGEGEBENE PERIPHERIEGERAET WIRD IN DIE LISTE DER AKTIVEN GERAETE (ACTIVE DEVICE TABLE, ADT) EINGETRAGEN. DER GERAETENAME KANN ANSCHLIESSEND ALS ZUSAETZLICHE SPEZIFIKATION FUER FILENAME EINGESETZT WERDEN. IST KEINE ADRESSE ANGEGEBEN, WIRD DER FILE, AUF DEN SICH DER GERAETENAME BEZIEHT, AUF DAS GEWAEHLTE GERAET LOKALISIERT UND GELADEN. VORAUSSETZUNG IST, DASS ES SICH UM EINEN 'DEVICE FILE' (TYPE=PROCEDURE, SUBTYPE=1) HANDELT. DER PROGRAMMNAME DARF NICHT 'NULL' SEIN UND DAS PROGRAMM DARF KEINE GESCHUETZTEN SPEICHERBEREICHE UEBERLAPPEN. DIE ZUWEISUNG DES DURCH DIE LADUNG BELEGTEN SPEICHERBEREICHES WIRD AUFRECHTERHALTEN (SIEHE ADT ODER KOMMANDO 'LADT'). WENN ENTRY=STARTADRESSE ANGEGEBEN (FILE VORHER LADEN !), WIRD DAS TREIBERPROGRAMM AUF DER ANGEGEBENEN ADRESSE GESTARTET. SIND DIE SPEICHERGRENZEN UNBEKANNT, WIRD DIE SPEICHERGROESSE NACH DEM ENTRY =0 GESETZT. SONST (OHNE ADRESSANGABE) WIRD EINE I/O-INITIALISIERUNG ZUM GERAET GESANDT, UM SPAETERE ANFORDERUNGEN ZU ERMOEGLICHEN. ACTIVATE BENUTZT ZWEI LOGISCHE GERAETE:

UNIT 0: TREIBERAUFRUF
UNIT 2: FEHLERMELDUNGEN

```
%ACTIVATE "PTAPE"
```

AKTIVIERT DEN LOCHBANDTREIBER 'PTAPE'. DER TREIBER IST SOMIT GELADEN; EINE INITIALISIERUNGSANFORDERUNG WURDE GESANDT. DIE EINTRAGUNG IN DIE ADT IST ERFOLGT.

7.2.2. ALLOCATE "UNTERE_GRENZE" "OBERE_GRENZE" "BLOCKLAENGE"

MIT DEM KOMMANDO WIRD VERSUCHT, VERFUEGBAREN SPEICHERBEREICH IN ANGEGEBENER LAENGE ZWISCHEN DEN ANGEGEBENEN GRENZEN ZU LOKALISIEREN.

- DIE "BLOCKLAENGE" WIRD AUFGERUNDET AUF 0 MODULO 80H (AUF DAS VIELFACHE VON 80H)
- DIE "UNTERE GRENZE" WIRD ABGERUNDET AUF 0 MODULO 80H
- DIE "OBERE GRENZE" WIRD AUFGERUNDET AUF OFFH MODULO 80H

DER SPEICHERBEREICH WIRD, FALLS NOCH FREI, IN DER MEMORY MAP ALS 'RESERVIERT' MARKIERT.
IST EINE RESERVIERUNG NICHT MOEGLICH, ERFOLGT DIE MELDUNG: 'INSUFFICIENT MEMORY' (GESCHUETZTER, D.H. BEREITS ANDERWEITIG BELEGTER SPEICHERBEREICH).

```
%A 7400 8000 380
```

DAS KOMMANDO BEWIRKT DIE SUCHE EINES 400H-SPEICHERBEREICHES (380H AUF VIELFACHES VON 80H AUFGERUNDET) IM ADRESSBEREICH VON 7400H BIS 807FH (8000H AUF OFFH MOD 80H AUFGERUNDET). ES ERFOLGT ENTWEDER DIE RESERVIERUNG IN DER MEMORY-MAP ODER EINE FEHLERMELDUNG.

7.2.3. BRIEF

UMSCHALTUNG DER KONSOLE AUF BRIEF-MODE (KURZ-BETRIEBSART). DIE WIEDERHOLUNG GEGEBENEN KOMMANDOS (SOGENANNTES ECHO) WIRD UNTERDRUECKT.
SIEHE AUCH: VERBOSE-KOMMANDO

```
%B
```

UMSCHALTUNG IN BRIEFMODE.

7.2.4. CAT ("STRING".OR.T="TYPE".OR.P="PROPS".OR.D="DRIVE"
.OR.F="FORMAT".OR.L="LISTINGDISPOSITION".OR.DATE^
"DATUM".OR.CDATE^"ERSTELLUNGSDATUM")*

AUSGABE ALLER FILENAMEN, DIE DIE ANGEGEBENEN BEDINGUNGEN
ERFUELLEN.
DIE REIHENFOLGE DER ANGABEN IST BELIEBIG.
IST KEINE OPTION ANGEGEBEN, WERDEN ALLE FILES (AUSSER DEN
'GEHEIMEN') AUSGEGEBEN, DIE IN ALLEN DIRECTORIES ALLER
AKTIVEN LAUFWERKE EXISTIEREN.
WERDEN GLEICHE OPTIONS MEHRMALS BENUTZT, IST DIE JEWEILS LETZTE
ANGABE GUELTIG.

"STRING": VOLLSTAENDIG ODER TEILWEISE SPEZIFIZIERTER FILENAME; IN DER ZEICHENKETTE WIRD DAS ZEICHEN '*' FUER ZEICHENFOLGEN BENUTZT, DIE BELIEBIG SEIN DUERFEN. Z.B.: 'A*.OLD' BEDEUTET: ALLE FILES, DIE MIT A BEGINNEN UND MIT .OLD ENDEN. ES KOENNEN MEHRERE, DURCH LEERZEICHEN GETRENNTE ZEICHENKETTEN ANGEGEBEN SEIN.

"FORMAT": WENN F=L : AUSGABE IM LANGEN FORMAT.
OHNE ANGABE: " " KURZEN "

"LISTING-
DISPOSITION": ANGABE DES GERAETES ODER FILES, AUF DAS ODER DEN DIE AUSGABE ERFOLGEN SOLL.
OHNE ANGABE: AUF KONSOLE

"DATUM": ANGABE DES DATUMS DER LETZTEN AENDERUNG.
FUER '^' KANN STEHEN:

=	AM	ANGEGEBENEN DATUM
<>	NICHT AM	"
>	NACH	"
<	VOR	"
>=	AM ODER NACH	"
<=	AM ODER VOR	"

"ERSTELLUNGS-
DATUM" : WIE DATUM, ABER ERSTELLUNGSDATUM ANSTELLE DATUM DER LETZTEN AENDERUNG.

"TYPE": FILETYPE; FOLGENDE SIND MOEGLICH:
A ASCII
P PROCEDURE
B BINARY
D DIRECTORY

"PROPS": PROPERTIES, D.H. FILE-EIGENSCHAFTEN:

W	WRITE PROTECTED, SCHREIBGESCHUETZT
E	ERASE PROTECTED; LOESCHGESCHUETZT
L	LOCKED, EIGENSCH. NICHT AENDERBAR
R	RANDOM, WAHLFREIER ZUGRIFF
S	SECRET, GEHEIM
F	FORCE (SIEHE KOMMANDO 'FORCE')
&	FILES MIT ALLEN EIGENSCHAFTEN, AUCH S

CAT BENUTZT DIE LOGISCHEN GERAETE 2 BIS 6:
UNIT 2: FEHLERMELDUNGEN
UNIT 3: AUSGABEGERAET (VOREINSTELLUNG)
UNIT 4: FUER DIRECTORY
UNIT 5: FUER DATEIEN, DIE IM DIRECTORY ENTHALTEN SIND
UNIT 6: AUSGABEGERAET, WENN OPTION "L=DATEINAME" SPEZIFIZIERT IST

```
%CAT F=L * .S * .L P=E L=CAT.LIST CDATE<810401
```

DIE NAMEN ALLER FILES, DIE MIT .S ODER .L ENDEN, DIE AUSSERDEM LOESCHGESCHUETZT SIND UND VOR DEM 1.APRIL 81 ERSTELLT WURDEN, WERDEN ALS FILE MIT DEM NAMEN CAT.LIST AUF DEM MASTER-DEVICE IM LANGEN FORMAT ABGELEGT.

7.2.5. CLOSE *.OR."UNIT"

ERZEUGT EINEN CLOSE-REQUEST FUER EINE LOGISCHE EINHEIT (ANGABE DER EINHEIT ALS HEXZAHL).
WIRD * ANSTELLE DER EINHEIT ANGEGEBEN, WERDEN ALLE EINHEITEN ABGESCHLOSSEN.
FEHLERMELDUNGEN WERDEN UNTERDRUECKT.

```
%CLOSE 5
```

BEWIRKT UEBERGABE EINES CLOSE-REQUESTS AN EINHEIT 5

7.2.6. COMPARE "FILE1" "FILE2"

DIE INHALTE DER FILES MIT DEN ANGEGEBENEN NAMEN WERDEN VERGLICHEN (MIT AUSNAHME DES DESCRIPTOR-RECORDS).
BEI IDENTITAET ERFOLGT KEINE TEXTAUSGABE.
BEI JEDEM FEHLERHAFTEN BYTE WIRD EINE FEHLERMELDUNG FOLGENDER FORM AUSGEGEBEN:

```
1: BYTE 01FC RECORD 0003 - B6  2: BYTE 01FC RECORD 0003 - A6
```

DIE KOMMANDOAUSFUEHRUNG KANN MIT DER TASTE ET2 ABGEBROCHEN WERDEN.
COMPARE BENUTZT FOLGENDE LOGISCHE GERAETE:
UNIT 2: FEHLERMELDUNGEN
UNIT 6: EINGABE FILE1
UNIT 7: EINGABE FILE2
BEACHTE: ES IST NICHT MOEGLICH, EINEN FILE MIT SICH SELBST ZU VERGLEICHEN.

```
%COMPARE "MYFILE" "YOURFILE"
%
```

DIE FILES DER ANGEGEBENEN NAMEN WURDEN BYTEWEISE VERGLICHEN.
DA KEINE FEHLERMELDUNG ERFOLGTE, LIEGT IDENTITAET VOR.

7.2.7. COPY "FILE 1" "FILE 2" (A.OR.U.OR.O.OR.RL="RECORD-LAENGE".OR.T="TYPE")

DER FILE MIT DEM NAMEN "FILE 1" WIRD MIT EINEM READ-BINARY-REQUEST GELESEN, DER INHALT WIRD IM FILE MIT DEM NAMEN "FILE 2" MIT EINEM WRITE-BINARY-REQUEST ABGELEGT.
"FILE 1" UND "FILE 2" KOENNEN FILENAMEN ODER VOLLSTAENDIGE FILE-SPEZIFIKATIONEN SEIN.
FILE-ATTRIBUTE VON "FILE 1" WERDEN AUF "FILE 2" UEBERTRAGEN.

-AUSNAHME: DAS DATUM DER LETZTEN AENDERUNG WIRD MIT DEM SYSTEMDATUM UEBERSCHRIEBEN.

-AUSNAHME: RECORDLAENGE, SOFERN DIESE MIT RL=80, 100, 200, 400 ODER 800 NEU FESTGELEGT WIRD.

-AUSNAHME: TYP, SOFERN DIESER DURCH D (DIRECTORY), B (BINARY), A (ASCII) ODER P (PROCEDURE) NEU FESTGELEGT WIRD.

-OPTION A (APPEND) = ANHAENGEN AN FILE2
- U (UPDATE) = EINFUEGEN AM ANFANG VON FILE 2
- O (OUTPUT) = UEBERSCHREIBEN VON FILE 2

COPY BENUTZT FOLGENDE LOGISCHE GERAETE:
UNIT 2: FEHLERMELDUNGEN
UNIT 6: QUELLFILE
UNIT 7: ZIELFILE

```
%COPY ¤ZDOS:2/THE.FILE ¤MYDOS/ANOTHER.FILE  RL=400
```

DER FILE 'THE.FILE' VON DRIVE 2 WIRD UNTER DEM NAMEN 'OTHER.FILE' AUF DAS GERAET ¤MYDOS ABGELEGT.
RECORD-LAENGE DES DESTINATION-FILES (ZIEL-FILE) = 400H
SOFERN DER FILE 'OTHER.FILE' SCHON EXISTIERT, WIRD DESSEN INHALT UEBERSCHRIEBEN.

```
%COPY NOTE.TO.SD ¤SD
```

DER FILE 'NOTE.TO.SD' WIRD AUF DEM MASTERGERAET (¤ZDOS) GESUCHT UND ZUM TREIBER ¤SD KOPIERT; ANDERS GESAGT, DIE TEXTDATEI 'NOTE.TO.SD' WIRD GEDRUCKT. DER TREIBER ¤SD MUSS VORHER AKTIVIERT WORDEN SEIN.

7.2.8. COPY.DISK ["S_DRIVE" TO "D_DRIVE"] [V]

DER INHALT DER DISKETTE IM GERAET "S_DRIVE" (QUELL-LAUFWERK=0, WENN NICHT ANDERS ANGEGEBEN) WIRD AUF DIE DISKETTE IM GERAET "D_DRIVE" (ZIEL-LAUFWERK=1, WENN NICHT ANDERS ANGEGEBEN) KOPIERT.
NACH DER KOMMANDOEINGABE ERFOLGT DIE FRAGE 'DRIVES READY?'. QUELL- UND ZIELDISKETTE BRAUCHEN ERST NACH DIESER MELDUNG GESTECKT WERDEN.
- TASTENEINGABE 'Y' : DISKETTE WIRD KOPIERT
- ALLE ANDEREN TASTEN: PROGRAMMABBRUCH.

```
%COPY.DISK
```

DISKETTE VON LAUFWERK 0 WIRD AUF LAUFWERK 1 KOPIERT.

ANGABE VON V IN DER KOMMANDOZEILE ---► BEIM KOPIERVORGANG WIRD ZUSAETZLICH EIN KONTROLLESEN DURCHGEFUEHRT (VERIFICATION).

```
%COPY.DISK 2 TO 0 V
 DRIVES READY? Y
 VERIFICATION COMPLETE
```

KOPIERT DIE DISKETTE VOM LAUFWERK 2 AUF DISKETTE IN LAUFWERK 0 MIT ZUSAETZLICHEM KONTROLLESEN.

7.2.9. DATE [YYMMDD]

OHNE ANGABE YYMMDD: ANZEIGE DES SYSTEMDATUMS, WELCHES ALS:
'DATE OF CREATION' =ERSTELLUNGSDATUM BZW.
'DATE OF LAST MODIFICATION'=DATUM DER LETZTEN AENDERUNG IN FILE-DESCRIPTOREN EINGETRAGEN WIRD.
MIT ANGABE DES DATUMS WIRD DAS SYSTEMDATUM (SPEICHERZELLEN 0FA2-0FA7) AUF DEN ANGEGEBENEN WERT GESETZT.
YY = JAHR
MM = MONAT 01..12
DD = TAG 01..31

```
%DATE 810519
THURSDAY, MAY 19, 1981
```

SETZT DAS AKTUELLE SYSTEMDATUM UND GIBT DIESES MIT WOCHENTAG AUF DISPLAY (UNIT 2) AUS.

7.2.10. DEACTIVATE "GERAETENAME"

LOESCHT DEN GERAETENAMEN VON DER LISTE DER AKTIVEN GERAETE (ACTIVE DEVICE TABLE, ADT). FUER DIE LOG. EINHEITEN, DIE DEM GERAETENAMEN ZUGEORDNET SIND, WIRD EIN I/O-CLOSE-REQUEST UND FUER DEN GERAETETREIBER SELBST EIN DEACTIVATE-CLOSE-REQUEST ERZEUGT. DER FUER DEN GERAETETREIBER BELEGTE SPEICHERBEREICH WIRD FREIGEGEBEN.
DIE DEACTIVIERUNG DES LETZTEN AKTIVEN GERAETES WIRD VERHINDERT, DA SONST KEINE QUELLE FUER EXTERNE KOMMANDOS MEHR EXISTIEREN WUERDE. DAS MASTER-DEVICE KANN NICHT DEACTIVIERT WERDEN.
FEHLERMELDUNGEN WERDEN AUF UNIT 2 AUSGEGEBEN.

```
%DEACTIVATE "PTAPE"
```

LOESCHT DEN TREIBER 'PTAPE' VON DER ACTIVE-DEVICE-TABLE UND ERZEUGT EIN CLOSE-REQUEST FUER ALLE EINHEITEN, DIE PTAPE ZUGEORDNET SIND. ES WIRD EIN DEACTIVATE-REQUEST ZU PTAPE GESANDT; DER HIERFUER RESERVIERTE SPEICHERPLATZ WIRD FREIGEGEBEN.

7.2.11. DEALLOCATE "BLOCKADRESSE" "BLOCKLAENGE"

MARKIERT EINEN SPEICHERBLOCK ANGEGEBENER LAENGE AB DER ANGEGEBENEN ADRESSE ALS NICHT RESERVIERT (FREI) IN SYSTEM-MEMORY-MAP (SPEICHERBELEGUNGSPLAN).
WENN DER VORGEGEBENE BLOCK NICHT VOLLSTAENDIG RESERVIERT WAR, ERSCHEINT DIE MELDUNG:

```
MEMORY PROTECT VIOLATION
(SPEICHERSCHUTZ-VERLETZUNG)
```

DIE "BLOCKADRESSE" WIRD ABGERUNDET AUF DAS VIELFACHE VON 80H,
DIE "BLOCKLAENGE" WIRD AUFGERUNDET AUF DAS VIELFACHE VON 80H.

```
%DEA 502F 13B5
```

LOESCHT DIE RESERVIERUNG DES SPEICHERBEREICHES
AB ADRESSE 5000H (502F ABGERUNDET)
MIT DER LAENGE 1400H (13B5 AUFGERUNDET).
DER BLOCK MUSS VORHER RESERVIERT GEWESEN SEIN.

7.2.12. DEBUG

EINSPRUNG IN DAS SYSTEMKERN-INTERNE MONITORPROGRAMM.
IN DIESEM PROGRAMM STEHEN KOMMANDOS FUER DIE PROGRAMMTESTUNG, SPEICHER-UND REGISTERANZEIGE BZW AENDERUNG, PORTEIN-UND AUSGABE, SOWIE PHYSISCHE DISKETTENARBEIT ZUR VERFUEGUNG.

Q
VERLASSEN DES DEBUGGERS INS AUFRUFENDE PROGRAMM. DER AUFRUF KANN VOM OS AUS ERFOLGEN. ER IST ABER AUCH VON JEDEM ANDEREN PROGRAMM (UEBER DIE ADRESSE OBFAH) MOEGLICH, WENN IN DEN SPEICHERZELLEN OFB5/OFB6 DIE RUECKKEHRADRESSE UND BIT 5 DER SPEICHERZELLE OFC4H GESETZT WURDE.

O
START DES LADE- UND INITIALISIERUNGSVORGANGES DES LOGISCHEN TEILS DES BETRIEBSSYSTEMS UND UEBERGANG IN DIESES.

R ["REGISTERNAME"]
ANZEIGE ALLER INTERNEN REGISTER ODER ANZEIGE EINES UND MOEGLICHKEIT SEINER MODIFIKATION.

D ("ADRESSE" ["LAENGE"])
ANZEIGE DES INHALTES EINER SPEICHERZELLE UND MOEGLICHKEIT DER NEUEINGABE ODER DARSTELLUNG EINES BEREICHES MIT ENTSPRECHENDEN PARAMETERN.

* DAS R-KOMMANDO (EINZELREGISTER) UND DAS D-KOMMANDO (SPEICHERZELLE) KANN MIT "Q" VERLASSEN WERDEN. DIE EINGABE VON ET1 BEWIRKT WEITERSCHALTUNG OHNE AENDERUNG.

M ("ZIELADRESSE" "QUELLADRESSE" "LAENGE")
UMSPEICHERN EINES BEREICHES ENTSPRECHEND DEN PARAMETERN.

PO ("PORTADRESSE" "BITMUSTER")
AUSGABE DES ANGEGEBENEN BITMUSTERS AUF DEN PORT.

PI ("PORTADRESSE")
EINGABE VOM PORT UND MOEGLICHKEIT EINER AUSGABE.

B ["PRUEFPUNKT"]
PRUEFPUNKTREGISTER LOESCHEN BZW. MIT ENTSPRECHENDER ADRESSE LADEN.

J ["ADRESSE"]
SPRUNG ZU ANGEGEBENER ODER IM PC-REGISTER EINGETRAGENER ADRESSE. IM STACK WIRD VORHER RUECKKEHRADRESSE EINGETRAGEN. DER SYSTEMSTATUS IST EINGESCHALTET:
SP=OCFEH I=OFH INTERRUPTMODE=2 INTERRUPT ENABLE
DIE RUECKKEHR IST UEBER "RETURN" ("RST38", NMI) MOEGLICH.

G ["ADRESSSE"]
SPRUNG ZUR ANGEGEBENEN BZW. IM PC-REGISTER EINGETRAGENEN ADRESSSE, LADEN DER REGISTER MIT DEM IM HAUPTSPEICHER ABGELEGTEN ABBILD UND EINTRAGEN DES VORHER EINgEGEBENE PRUEFPUNKTES.
DIE RUECKKEHR IST SONST UEBER "RST38" ODER NMI MOEGLICH.

L ("DISKADRESSE" "HS-ADRESSE" "LAENGE")
LESEN EINES RECORDS (MAX. 0D00H BYTES, 1...26 SECTOREN).

S ("DISKADRESSE" "HS-ADRESSE" "LAENGE")
SCHREIBEN EINES RECORDS AUF DISKETTE.
DISKADRESSE: H-BYTE SPURNUMMER 0...4C (28)
L-BYTE BIT 0..4 SECTORNUMMER 0...19
5..7 LAUFWERKSNUMMER
ACHTUNG: ALLE SEKTOREN, DIE GESCHRIEBEN WERDEN, ERHALTEN DIE GLEICHEN ZEIGERINFORMATIONEN. ALSO IMMER NUR E I N E N RECORD (DAS KOENNEN MEHRERE SEKTOREN SEIN) MIT EINEM S-KONMANDO SCHREIBEN

```
%D
D
+R                  ;REGISTERANZEIGE
A  B  C  D  E  F  H  L  I   IX   IY   PC   SP
00 00 00 00 00 00 00 00 00 0000 0000 0000 0000
00 00 00 00 00 00 00 00
D FCA               ;ANZEIGE DER LAUFWERKSBELEGUNG
0FCA 16 Q           ;0=8"; 1,2=5"; 3=NICHT VORHANDEN
+Q
%
```

7.2.13. DEFINE ("UNIT/FILENAME".OR."UNIT/DEVICENAME".OR."UNIT"*
.OR.*)+[A .OR. O .OR. U .OR. I .OR. NP .OR. NO]

VERBINDET EINE LOGISCHE EINHEIT (BENANNT DURCH ZAHL VON 1 BIS 20) MIT EINEM DER ZUR ZEIT AKTIVEN GERAETE ODER GIBT DIE EINHEIT ZURUECK ENTSPRECHEND DES ZUSTANDES BEI SYSTEMINITIALISIERUNG.
DIE EINHEITEN 1,2,3 KOENNEN SYMBOLISCH BENANNT WERDEN ALS: CONIN, CONOUT, SYSLST.
WENN DIE EINHEIT VORHER DEFINIERT WAR, WIRD EIN CLOSE-REQUEST ERZEUGT. EIN FILENAME KANN OPTIONAL ZUR EINHEIT HINZUGEFUEGT WERDEN. NACH DEFINE-KOMMANDO KOENNEN FUER DIE ANGEGEBENEN EINHEITEN ASSIGN- UND OPEN-REQUESTS ERZEUGT WERDEN.

WENN IM DEFINE-KOMMANDO FUER UNIT EIN '*' STEHT, WERDEN ALLE LOGISCHEN GERAETE AUF IHREN ANFANGSWERT, WIE ER NACH SYSTEMSTART EINGENOMMEN WIRD, ZURUECKDEFINIERT.
OPTIONS (SIEHE AUCH ZDOS-BESCHREIBUNG, OPEN-REQUEST):
A MIT OPEN-REQUEST FUER APPEND
O MIT OPEN-REQUEST FUER OUTPUT
U MIT OPEN-REQUEST FUER UPDATE
I MIT OPEN-REQUEST FUER INPUT
NP MIT OPEN-REQUEST FUER NEW FILE
NO DEFINE O H N E OPEN-REQUEST

```
%DEFINE 12 =YOUR.DOS/YOURFILE NO
```

ORDNET DER LOG. EINHEIT 12 DAS GERAET YOUR.DOS ZU (WELCHES AKTIVIERT SEIN MUSS).
DANN WIRD EIN ASSIGN-I/O-REQUEST ERZEUGT MIT DEM FILENAMEN 'YOURFILE', ABER KEIN OPEN-REQUEST AUSGESANDT (OPTION 'NO').

```
%DEFINE *
```

ALLE EINHEITEN ERHALTEN EINEN CLOSE-REQUEST UND WERDEN AUF IHREN VOREINGESTELLTEN WERT (NACH SYSTEMLADEN) ZURUECKDEFINIERT.

7.2.14. DELETE ("STRING" .OR. T="TYPE" .OR. P="PROPS" .OR. D="DRIVE" .OR. Q="QUERY" .OR. DATE^"DATUM" .OR. CDATE^"DATUM")*

"STRING" IST EIN VOLLSTAENDIG ODER TEILWEISE ANGEGEBENER FILENAME (WIE BEI KOMMANDO 'CAT').
DAS KOMMANDO LOESCHT DIE DURCH DIE ZEICHENKETTE SPEZIFIZIERTEN FILENAMEN IM DIRECTORY UND LOESCHT DIE BELEGUNG DER BENUTZTEN RECORDS.
OHNE ANGABE VON OPTIONS - ALLE (NICHT GEHEIMEN) FILES IN ALLEN AKTIVIERTEN GEBIETEN WERDEN GELOESCHT.
WERDEN AUSSER "STRING" ANDERE OPTIONS MEHRFACH EINGEGEBEN, IST DIE LETZTE ANGABE GUELTIG.
NACH KOMMANDOEINGABE ERFOLGT DIE FRAGE (QUERY):
        %DELETE DRIVE/FILENAME (Y/N/A/Q)?
EINES DER ANGEGEBENEN ZEICHEN IST EINZUGEBEN:
  'Y' :JA - BENANNTEN FILE LOESCHEN
  'N' :NEIN-BENANNTEN FILE NICHT LOESCHEN
  'A' :ALLE FILES ENTSPRECHEND DER KOMMANDOEINGABE AUSSER DEN ZUVOR BEREITS ABGEFRAGTEN WERDEN GELOESCHT.
  'Q' :KEINE WEITEREN FILES LOESCHEN.

OPTION Q=N : BEDEUTET NO QUERY (FRAGE UNTERDRUECKEN). ALLE NICHTGEHEIMEN FILES WERDEN GELOESCHT.
SONSTIGE OPTIONS: SIEHE BEI KOMMANDO 'CAT'

```
        %DELETE D=1 P=R *.BASIC
```

LOESCHT ALLE 'RANDOM'-FILES AUF DRIVE 1, DEREN NAMEN MIT '.BASIC' ENDEN.

7.2.15. DISPLAY

ABBILDUNG DES SPEICHERBELEGUNGSPLANES AUF DIE SYSTEMCONSOLE.
(JE EINE ZEILE FUER JE 1000H BYTES)
'A' = ALLOCATED (RESERVIERT)
'.' = FREI

7.2.16. DO "KOMMANDOFILE" (PARAMETER)*

AUSFUEHRUNG DER KOMMANDOS, DIE IM KOMMANDOFILE STEHEN.
SUBSTITUTION DES N-TEN PARAMETERS FUER JEDES #N, WOBEI N <= DER PARAMETERANZAHL IST.
DIE MOEGLICHE PARAMETERANZAHL WIRD FESTGELEGT DURCH PAARWEISE BENUTZUNG DER ECKIGEN KLAMMERN []. PARAMETER UEBER DIE ANZAHL DER KLAMMERPAARE HINAUS WERDEN IGNORIERT.

DAS DO KOMMANDO KANN REKURSIV BENUTZT WERDEN, D.H., AUCH IM KOMMANDOFILE SELBST, WENN DIE PROPERTIES VON DO "P" ENTHALTEN.
IM BEISPIEL SOLL EIN KOMMANDOFILE NAMENS 'DRUCK' MIT DEM INHALT:

```
        ACTIVATE #DARO
        [COPY #1 #DARO[;COPY #2 #DARO[;COPY #3 #DARO]]]
        DEACTIVATE #DARO
```

EXISTIEREN.

```
        %DO DRUCK FILE1 FILE2 FILE3
```

DRUCKER DARO WIRD ACTIVIERT.
DIE FILES MIT DEN NAMEN FILE1, FILE2, FILE3 WERDEN GEDRUCKT;
DRUCKER DARO WIRD DEACTIVIERT.

7.2.17. DUMP FILENAME [M[N]]

VOM ANGEGEBENEN FILE WIRD EIN SPEICHERBELEGUNGSBILD (HEXADEZIMAL UND ASCII) AUF SYSLST AUSGEGEBEN.
NICHTDRUCKBARE BYTES WERDEN IN DER ASCII-SPALTE ALS '.' DARGESTELLT.

M = NUMMER DES 1. AUSZUGEBENDEN RECORDS
N = " " LETZTEN AUSZUGEBENDEN RECORDS
WENN DIE ANGABEN M,N FEHLEN, WIRD VOM 1. BIS ZUM LETZTEN RECORD AUSGEGEBEN. (ZAHLENANGABE DEZIMAL).
DAS KOMMANDO BENUTZT DIE LOGISCHEN GERAETE:
UNIT 3: AUSGABE DES SPEICHERBELEGUNGSBILDES
UNIT 4: EINGABE DER DARZUSTELLENDEN DATEI

```
%DUMP =MICRO.80:2/DATA
```

DER INHALT DES FILES "DATA" DES GERAETES "MICRO.80", DRIVE 2 WIRD HEXADEZIMAL UND ASCII AUF DAS SYSTEM-AUSGABE-GERAET AUSGEGEBEN.

7.2.18. ECHO "ZEICHENKETTE"

KOPIERT DIE ZEICHENKETTE AUF DIE KONSOLE (UNIT 2).
ERMOEGLICHT TEXTAUSGABEN MITTELS KOMMANDOZEILEN.

```
%COPY,;ECHO DISKETTEN EINLEGEN; PAUSE;I;X
```

NACHDEM DAS KOMMANDO COPY GELADEN WURDE, WIRD DIE ZEICHENKETTE 'DISKETTEN EINLEGEN' AUF DEN BILDSCHIRM (GENAUER: UNIT 2) AUSGEGEBEN UND DIE KOMMANDOABARBEITUNG WIRD UNTERBROCHEN (PAUSE), BIS DER BEDIENER EINE TASTE BETAETIGT. ANSCHLIESSEND WIRD DAS INTERNE KOMMANDO I ABGEARBEITET UND MIT X DAS BEREITS GELADENE 'COPY' GESTARTET.

7.2.19. EXTRACT "FILENAME"

VOM FILE MIT ANGEGEBENEN NAMEN WERDEN FOLGENDE DATEN AUF UNIT 2 AUSGEGEBEN:

- RECORD COUNT (ANZAHL DER RECORDS)
- RECORD LENGTH (RECORD-LAENGE)
- BYTES IN LAST RECORD (BYTEANZAHL IM LETZTEN RECORD)

BEI PROCEDURE-FILES ZUSAETZLICH:

- ENTRY POINT (PROGRAMMEINTRITTSPUNKT, STARTADRESSE)
- LOW ADRESS (NIEDRIGSTE BENUTZTE SPEICHERADRESSE)
- HIGH ADRESS (HOECHSTE BENUTZTE SPEICHERADRESSE)
- STACK SIZE (STACKGROESSE FUER DAS PROGRAMM)
- SEGMENTS (ADRESSEN DER SPEICHERSEGMENTE, DIE VOM FILE BELEGT WERDEN)

'EXTRACT' KANN BENUTZT WERDEN, UM DIE GUENSTIGSTE RECORDLAENGE FUER PROCEDUREFILES ZU FINDEN.

```
%EXTRACT EXTRACT

RECORD COUNT=0001 RECORD LENGTH=0200 BYTES IN LAST RECORD=0200
ENTRY=4000  LOW ADDRES=4000  HIGH ADDRESS=43FF STACK SIZE=0080
SEGMENTS:
4400 45F2
```

7.2.20. FORCE "KOMMANDO" "PARAMETERLISTE"

BEWIRKT, DASS ALLE KOMMANDOFILES IN DER LAUFENDEN KOMMANDOZEILE (BIS ';' ODER ZEILENENDE) GELADEN WERDEN, OHNE RUECKSICHT AUF VORHERIGE SPEICHERRESERVIERUNG.
(NORMALERWEISE WIRD EIN PROCEDUREFILE NUR GELADEN, WENN DER ERFORDERLICHE SPEICHERBEREICH NICHT RESERVIERT IST.)
ANWENDUNGSBEISPIELE FUER FORCE: OVERLAYSTRUKTUREN UND REKURSIVE PROGRAMMAUFRUFE.

WENN EIN FILE DIE EIGENSCHAFT 'F' BESITZT (PROPS = F, FORCE MEMORY ALLOCATION), HAT DIES DIE GLEICHE AUSWIRKUNG WIE DAS FORCE-KOMMANDO.

```
%F "FILEA","FILEB",;"FILEC"
```

DIE PROCEDUREFILES "FILEA" UND "FILEB" WERDEN GELADEN UND NICHT AUSGEFUEHRT.
"FILEC" WIRD GELADEN UND NUR DANN AUSGEFUEHRT, WENN DER BENOETIGTE SPEICHER NICHT RESERVIERT WAR.

7.2.21. FORMAT

FORMATIERT EINE 8"- ODER 5,25"-DISKETTE MIT 77 BZW 40 SPUREN ZU JE 26 SECTOREN IM UDOS-IBM-FORMAT.
DER DISKETTENBELEGUNGSPLAN (DISK ALLOCATION MAP)
UND EINE DISKETTEN-NUTZUNGS-STATISTIK WERDEN INITIALISIERT.
EIN LEERES DIRECTORY (DISKETTENINHALTSVERZEICHNIS), DAS NUR DIE ANGABEN DES DIRECTORY-FILES SELBST ENTHAELT, WIRD ANGELEGT.

3 SECTOREN WERDEN VOM DISK.-BELEGUNGSPLAN (DISK ALLOCATION MAP) UND 11 SECTOREN WERDEN VOM DIRECTORY BELEGT.

BEIM FORMATIEREN EINER SYSTEMDISKETTE WERDEN ZUSAETZLICH FUER DEN SYSTEMLADER 9, DAS PHYSISCHE BETRIEBSSYSTEM 26 UND DEN BOOTSTRAPLADER 6 SECTOREN RESERVIERT.

NACH DER ERZEUGUNG DES PHYSISCHEN AUFZEICHNUNGSFORMATES WERDEN ALLE SPUREN KONTROLLGELESEN. BEI FESTGESTELLTEN AUFZEICHNUNGSFEHLERN (DISKETTENDEFEKTE) WERDEN DIE DEFEKTEN SPUREN IM DISKETTENBELEGUNGSPLAN ALS BELEGT GEKENNZEICHNET, WODURCH EINE WEITERE BENUTZUNG VERHINDERT WIRD. DIE DISKETTE IST ABER WEITERHIN VERWENDBAR.

ZUR FORMATIERUNG EINER SYSTEMDISKETTE MUSS EIN ENTSPRECHENDES ORIGINAL (5,25" BZW. 8") IN DEM LAUFWERK (X), VON DEM DAS SYSTEM BEIM START GELADEN WURDE, ZUR VERFUEGUNG STEHEN.
WENN KEINE SYSTEMDISKETTE IN DRIVE X STECKT, SIND FOLGENDE ANWEISUNGEN AUSZUFUEHREN:

```
"INSERT SYSTEMDISK IN DRIVE X      READY?:"
(STECKE SYSTEMDISKETTE IN OS-DRIVE, BETAETIGE "Y")
"REPLACE FORMATET DISK IN DRIVE X READY?:"
(FORMATIERTE DISKETTE ZURUECK IN DRIVE X. DRUECKE: "Y")
```

```
%FORMAT
 SYSTEMDISK? Y
 DRIVE? 1
 ID? UDOS.SYSTEM
 READY? Y
```

(MUSS MIT 'Y' BEANTWORTET WERDEN, SONST ABBRUCH!)
FORMATIERT DIE DISKETTE IN LAUFWERK 1 ALS SYSTEMDISKETTE MIT DEN ANGEGEBENEN NAMEN.

7.2.22. HELP ("KOMMANDO".OR.*)*

```
%HELP
```

BESCHREIBUNG DES HELP-KOMMANDOS.

```
%HELP *
```

AUFLISTUNG ALLER KOMMANDOS, FUER DIE EIN 'HELP' EXISTIERT.

```
%HELP "KOMMANDO"
```

BESCHREIBUNG DES ANGEGEBENEN KOMMANDOS (AUF UNIT 2; DER BESCHREIBUNGSTEXT WIRD VON UNIT 4 GEHOLT!).

SIEHE AUCH: HELP ASCII, HELP ERROR.
ERLAEUTERUNG DER SCHREIBWEISE IN ALLEN HELP-BESCHREIBUNGEN:
DIE ZEICHEN, DIE ANWENDERSPEZIFISCH EINZUSETZEN SIND (FILENAMEN, ZAHLEN USW.), WERDEN IN "..." EINGESCHLOSSEN.
DIESE ZEICHEN DIENEN NUR ZUR KENNZEICHNUNG ALS VARIABLE AUSDRUECKE UND WERDEN N I C H T MIT EINGEGEBEN.
IN ECKIGEN KLAMMERN [...] SIND OPTIONS (NICHT UNBEDINGT NOTWENDIGE PARAMETER) ANGEGEBEN.
ALS ZEICHEN FUER DAS LOGISCHE ODER WIRD .OR. VERWENDET.

7.2.23. IMAGE "FILENAME" ("ERSTE SPEICHERADRESSE" "LETZTE SPEICHERADRESSE")+[E="EINTRITTSPUNKT"] [RL="RECORD-LAENGE"] [ST="STACKTIEFE]

KOPIERT DEN/DIE ANGEGEBENEN SPEICHERBEREICH(E) AUF DEN PROCEDUREFILE MIT DEN ANGEGEBENEN NAMEN (SUBTYPE=0)
ALLE ZAHLEN SIND HEXADEZIMAL ANZUGEBEN.
WENN ANGABE 'E' FEHLT: "EINTRITTSPUNKT" = 0
"RECORD LAENGE" DARF SEIN 80H, 100H, 200H, 400H ODER 800H
OHNE ANGABE: RL = 80H
"STACKTIEFE" - OHNE ANGABE = 80H

MINDESTENS EIN - HOECHSTENS 16 SEGMENTE KOENNEN SPEZIFIZIERT WERDEN. BEIM SCHREIBEN DES FILES WERDEN DIE EXAKTEN SPEICHERBELEGUNGEN EINSCHLIESSLICH ERSTE UND LETZTE SPEICHERZELLE FUER JEDES SEGMENT KOPIERT.
HOECHSTE UND NIEDRIGSTE SPEICHERZELLE DES FILES WERDEN IN DESCRIPTORRECORD ABGELEGT.
BEIM LADEN DES FILES WERDEN DIESE ADRESSEN FUER DIE SPEICHERRESERVIERUNG (ALLOCATION) EINGESETZT.

```
%IMAGE ZWEI.BLOECKE 4400 4425 7000 7FF0 E=7000
```

KOPIERT DEN INHALT DER SPEICHERBEREICHE 4000-4425 UND 7000-7FF0 AUF DEN FILE NAMENS 'ZWEI.BLOECKE'.
DIESER FILE ENTHAELT 33 RECORDS JE 80H BYTES,
EINTRITTSPUNKT = 7000H
STACK-TIEFE = 80H

7.2.24. INITIALIZE [GERAETENAME[PARAMETERLISTE]]

SENDET EIN INITIALIZE-REQUEST ZUM MASTER-DEVICE ODER ZU EINEM OPTIONAL ANGEGEBENEN GERAET, WELCHES VORHER AKTIVIERT SEIN MUSS (DAS GERAET WIRD 'ANGEWIESEN' ODER 'INITIALISIERT').

DIE ZUSATZ-PARAMETERADRESSE DES VECTORS (SPV, SUPPLEMENTAL PARAMETER VECTOR ADDRESS) ZEIGT AUF DAS TRENNZEICHEN HINTER DEN KOMMANDO, BZW., SOFERN ANGEGEBEN, HINTER DEN DEVICENAMEN.

```
%I ¤NY.VIDEO.DRIVER BUFFER=C000
```

SENDET EIN INITIALIZE-REQUEST ZUM GERAET 'MY.VIDEO.DRIVER', DIE ZUSATZPARAMETER-ADRESSE DES VECTORS ZEIGT IN KOMMANDO-PUFFER VOR DIE ZEICHENKETTE 'BUFFER...'.

```
%I
```

SENDET INITIALIZE-REQUEST ZUM MASTER-GERAET (¤ZDOS). DABEI WERDEN VON ALLEN AKTIVEN FLOPPY-DISK-LAUFWERKEN DIE DISKETTENBELEGUNGSMAPS EINGELESEN UND IM ARBEITSSPEICHER BEREITGESTELLT. DER GESAMTE ANWENDERSPEICHER WIRD FREIGEGEBEN (SIEHE AUCH RELEASE-KOMMANDO).

7.2.25. LADT

LISTET DIE ZUR ZEIT AKTIVEN GERAETEPROGRAMME, DEREN EINTRITTSPUNKTE, GROESSE UND DIE ZUORDNUNG ZU DEN LOGISCHEN EINHEITEN (AUF UNIT 3) AUF.

```
%LADT
LADT
  DEVICE          ENTRY   MODUL       MAPPED
  NAME            POINT   ADDRESS     UNITS
ZDOS               2600    1A00    0  4  5  6  7  8  9 10
                                   11 12 13 14 15 16 17 18
                                   19
CON                2110    0500    1  2  3
NULL               1D51    0000   20
PCON               0BBE    0000
FLOPPY             0BFD    0000
```

'MODUL ADDRESS =0' BEDEUTET, DASS DIESE TREIBER ZUM SYSTEM GEHOEREN UND NICHT AKTIVIERT WERDEN MUESSEN.

7.2.26. MASTER [GERAETENAME]

OHNE OPTION: AUSGABE, WELCHES GERAET Z.Z. MASTER_DEVICE IST.

MIT OPTION: DAS BENANNTE GERAET (VORHER ZU AKTIVIEREN) WIRD ZUM MASTER_DEVICE, D.H., ZUR QUELLE(ZIEL) UNVOLLSTAENDIG SPEZIFIZIERTER FILENAMEN. DIES ERMOEGLICHT DEM ANWENDER, MEHRERE FILESYSTEME (Z.B. MAGNETBANDEINHEITEN ODER FESTPLATTE) GLEICHZEITIG ZU NUTZEN OHNE JEDESMAL DEN VOLLSTAENDIGEN FILENAMEN ANGEBEN ZU MUESSEN.

```
%MASTER ¤NEW.DOS
```

NEW.DOS WIRD ZUR QUELLE(ZIEL) VON FILES, BEI DENEN DIE ANGABE DES GERAETENAMENS FEHLT.

2.2.27. MOVE ("STRING" .OR. T="TYPE" .OR. P="PROPS" .OR.
F="FORMAT" .OR. D="DST.DEVICE" .OR. S="SOURCE.DEVICE"
.OR. L="LISTING_DISPOSITION .OR. Q="QUERY" .OR.
DATE^"DATUM" .OR. CDATE^"DATUM"

DIE DIRECTORY DES "SOURCE.DEVICE" (QUELL-GERAET) WIRD AUF "STRING" ABGESUCHT. DIE GEFUNDENEN FILES WERDEN ZUM "DESTINATION.DEVICE" (ZIEL-GERAET) KOPIERT, SOFERN DIE OPTIONS UEBEREINSTIMMEN. OHNE ANGABE VON ZIEL UND QUELLE: QUELLE = DRIVE 0, ZIEL = DRIVE 1.

"STRING" = VOLLSTAENDIG ODER TEILWEISE ANGEGEBENER FILENAME (FUER WEGGELASSENE BUCHSTABEN STEHT *). SIEHE AUCH BEI 'CAT'.

"FORMAT" F=L BEDEUTET LANGES FORMAT DER MOVE-LISTE

"LISTING_DISPOSITION" GERAET ODER FILE, AUF DAS/DEN DIE LISTE DER KOPIERTEN FILES DARGESTELLT WIRD. OHNE ANGABE: UNIT 3 =SYSLST (KONSOLE).

"QUERY" Q=Y BEDEUTET, DASS VOR JEDEM KOPIEREN EINE ABFRAGE ERFOLGT, DIE MIT Y,N,A ODER Q BEANTWORTET WERDEN KANN. (SIEHE BEI DELETE)
OHNE Q=Y : FRAGE WIRD UNTERDRUECKT.

OPTIONS T, P, DATE UND CDATE SIEHE BEI KOMMANDO CAT!
SOLLEN FILES MIT DER EIGENSCHAFT S=SECRET KOPIERT WERDEN, IST ALS OPTION P=& ANZUGEBEN ODER P=(A L L E EIGENSCHAFTEN DES FILES).

DAS KOMMANDO MOVE BENUTZT FOLGENDE LOGISCHE GERÄTE:

UNIT 3: VOREINGESTELLTES LISTGERÄT (FALLS KEIN 'L=' SPEZIFIZIERT WURDE)

UNIT 4: QUELLFILE

UNIT 5: ZIELFILE

UNIT 6: ÜBER 'L=...' GEWÄHLTES LISTGERÄT

%MOVE T=P SYS* L=¤DARO S=1 D=0 DATE>=810401

KOPIERT ALLE PROCEDURE-FILES, DEREN NAMEN MIT SYS BEGINNEN UND AM ODER NACH DEM 1. APRIL 81 ZULETZT GEAENDERT WURDEN, VON DRIVE 1 NACH DRIVE 0. AUF DARO-DRUCKER WIRD DIE LISTE DER KOPIERTEN FILES AUSGEGEBEN.

%MOVE*P=&S=1 D=¤NULL P=L

ALLE DATEIEN (DATEINAME *, MIT ALLEN EIGENSCHAFTEN P=&) VOM LAUFWERK 1 (QUELLE) WERDEN ZUM NULLTREIBER (¤ NULL) KOPIERT. ANGEZEIGT WIRD DABEI DAS LANGE FORMAT (SIEHE CAT). SOMIT WERDEN ALLE DATEIEN EINER DISKETTE GELESEN UND DABEI AUFTRETENDE FEHLER WERDEN ANGEZEIGT (ALS SOG. SOURCE-FILE READ-ERRORS). DAMIT STEHT EIN GEEIGNETES MITTEL ZUR EFFEKTIVEN ZEIGERKONTROLLE ZUR VERFÜGUNG. DIESES KOMMANDO SOLLTE ZUR KONTROLLE NACH JEDER KOPIEROPERATION ANGEWENDET WERDEN.

7.2.28. PAUSE

STAENDIGE ABFRAGE VON CONIN (UNIT 1, TASTATUR).

WENN ET2 EINGEGEBEN WIRD, WERDEN ALLE KOMMANDOS, DIE IN DER KOMMANDOZEILE NACH 'PAUSE' STEHEN, IGNORIERT.
WENN EINE ANDERE TASTE GEDRUECKT WIRD, WERDEN DIE IN DER ZEILE FOLGENDEN KOMMANDOS ABGEARBEITET.
(WIRD INSBESONDERE ANGEWANDT IN KOMMANDOFILES, DIE MIT 'DO' AUFGERUFEN WERDEN.)

FUER FOLGENDES BEISPIEL MUSS EIN FILE NAMENS 'MOVE.IT' EXISTIEREN, DER BEINHALTET:

```
MOVE,ECHO DISKETTEN STECKEN;PAUSE;I;X 4000

%DO MOVE.IT
```

MOVE WIRD GELADEN, ABER NOCH NICHT AUSGEFUEHRT.
ECHO WIRD GELADEN UND AUSGEFUEHRT (DISPLAYAUSGABE:'DISKETTEN STECKEN') PAUSE WARTET AUF TASTENEINGABE
NACH DEM STECKEN DER DISKETTEN WIRD TASTE ET2 ODER ANDERE GEDRUECKT.

BEI ET2:	ABBRUCH OHNE AUSFUEHRUNG VON MOVE
SONST :	MOVE AUSFUEHREN, D.H., ALLE NICHT GEHEIMEN FILES VON DRIVE 0 NACH DRIVE 1 KOPIEREN (4000 IST DIE STARTADRESSE DES MOVE-PROGRAMMES!!!).

7.2.29. PRINT "FILENAME"

MIT DIESEM KOMMANDO WIRD DIE DATEI MIT DEM ANGEGEBENEN NAMEN AUF DAS LOGISCHE GERAET SYSLST (STANDARDFALL: BILDSCHIRM) AUSGEGEBEN. DIE ANGEGEBENE DATEI MUSS EINE TEXTDATEI (DATEITYP A) SEIN.

```
%PRINT  OS.INIT
```

DIE KOMMANDODATEI OS.INIT WIRD AUF BILDSCHIRM AUSGEGEBEN.

WENN DER DRUCKER UEBER DIE MONITOR-TASTE (BZW. TASTE OFF) AKTIVIERT IST, WIRD DIE SPEZIFIZIERTE DATEI PARALLEL ZUR BILDSCHIRMAUSGABE NOCH GEDRUCKT.
%

7.2.30. RELEASE

BEIM LADEN VON PROCEDURE-FILES WIRD DER BENOETIGTE SPEICHERBEREICH RESERVIERT UND NACH AUSFUEHRUNG DES PROCEDURE-FILES WIEDER FREIGEGEBEN.
WENN EIN FILE GELADEN IST, ABER NICHT ALS EXTERNES KOMMANDO AUSGEFUEHRT WURDE, WIE DIES IM TEST MIT DEM DEBUGGER DER FALL IST, IST ES ERFORDERLICH, DEN RESERVIERTEN SPEICHERBEREICH WIEDER FREIZUGEBEN.
DAS RELEASE-KOMMANDO GIBT D E N SPEICHERBEREICH FREI, DER NACH DER AUSFUEHRUNG DES LETZTEN KOMMANDOS DURCH PROCEDURE-FILE LADUNG BELEGT WURDE.

```
%MOVE,                          ;FILE 'MOVE' LADEN
%STATUS                         ;STATUS SOLL GELADEN UND AUSGE-
                                FUEHRT WERDEN
MEMORY PROTECT VIOLATION          (BELEGUNG RESERVIERTEN
                                   BEREICHES)
%R                              ;SPEICHERBEREICH FREIGEBEN
%STATUS
DRIVE 0  DISK MUELLER
 256 SECTORS USED
 784 SECTORS AVAILABLE
%
```

7.2.31. RENAME ("OLDFILE" "NEWFILE" .OR. "DEVICE:DRIVE" ID="'NEW_DISK_NAME'")*

FILE UMBENENNEN: DER FILE MIT DEM NAMEN "OLDFILE" WIRD AUF DEN NAMEN "NEWFILE" UMBENANNT

DISKETTENNAME UMBENENNEN: DER NAME DER DISKETTE IM ANGEGEBENEN LAUFWERK (DRIVE) WIRD MIT DER ZWISCHEN APOSTROPH STEHENDEN ZEICHENKETTE UEBERSCHRIEBEN (DEVICE = ¤FLOPPY)
MAX. LAENGE DES 'NEW.DISK.NAME':
24 ZEICHEN FUER ¤FLOPPY
DER NAME DARF CARRIAGE RETURN, SEMIKOLON UND APOSTROPH NICHT ENTHALTEN.

```
%RENAME ¤MYDOS/FILE.X FILE.Y
```

BEWIRKT: ASSIGN-REQUEST, RENAME-REQUEST FUER GERAET ¤MYDOS, FILE.X WIRD IN FILE.Y UMBENANNT

```
%RENAME ¤FLOPPY:1 ID='NEW.WORKDISK'
```

BEWIRKT: UMBENENNUNG DES DISKETTENIDENTIFIKATORS IM LAUFWERK 1 UND ERZEUGUNG EINES INIT-REQUESTS.

7.2.32. RESTORE_TABS

ERSETZT DIE AKTUELLE 134-ZEICHEN-DISPLAY-TABULATOR-BELEGUNG DURCH DIE TAB'S IM ANGEGEBENEN FILE.
DER FILE MUSS DURCH DAS SAVE_TAB KOMMANDO ERZEUGT WORDEN SEIN.

```
%RESTORE_TABS TAB.7
```

ERSETZT DIE TAB'S IM TAB-SPEICHER (ADRESSE 24B0..2534) DURCH DIE TABS, DIE IM FILE TAB.7 FESTGELEGT SIND,

7.2.33. SAVE_TABS FILENAME

LEGT DIE Z.Z. AKTUELLE TABULATOR-BELEGUNG IM FILE "FILENAME" AB.

DIE TAB-BELEGUNG KANN DURCH DAS KOMMANDO SET TABSIZE=N GEAENDERT WERDEN.

```
%SAVE_TABS TAB.7
```

DIE AKTUELLE TAB-BELEGUNG WIRD IN FILE "TAB.7" ABGELEGT.

7.2.34. SET OPTION_LIST

SETZT VERSCHIEDENE SYSTEMPARAMETER ODER FILE-ATTRIBUTE.
OPTION_LIST KANN EINE BELIEBIGE KOMBINATION FOLGENDER PARAMETER SEIN (REIHENFOLGE BELIEBIG).

CHRDEL="C" — SYMBOL ZUM LOESCHEN EINES ZEICHENS.

LINDEL="C" — SYMBOL ZUM LOESCHEN EINER ZEILE

NULLCT="N" — ANZAHL DER NULL-ZEICHEN (ASCII-CODE= 00), DIE NACH JEDEM CARRIAGE RETURN ANGEHAENGT WERDEN (ZUR ANPASSUNG AN LANGSAME AUSGABEGERAETE BSP: FERNSCHREIBER)

SPEED="N" — BAUDRATE FUER SERIELLE DATENUEBERTRAGUNG (FOR 'SERIALCOMMUNICATION PORT')

LFCNT="N" — LINEFEED COUNT, ANZAHL DER ZEILENVORSCHUEBE NACH JEDEM CARRIAGE RETURN.

TABSIZE="N" — UMDEFINIEREN DER TAB'S AUF DAS VIELFACHE VON "N"

PROPERTIES OF "FILENAME" TO "PLIST"
SETZT EIGENSCHAFTEN DES FILES ENTSPRECHEND "PLIST" (SIEHE AUCH CAT)
SOLLEN DIE EIGENSCHAFTEN GELOESCHT WERDEN (AUSSER 'L'), IST '*' EINZUGEBEN!

SUBTYPE OF "FILENAME" TO "SUBTYPE"
SUBTYPE DES FILES SETZEN

TYPE OF "FILENAME" TO "TYPE"
FILETYP SETZEN

LOW_ADDRESS OF "FILENAME" TO "NN"
UNTERE PROGRAMMBELEGUNGSADRESSE SETZEN

HIGH_ADDRESS OF "FILENAME" TO "NN"
OBERE PROGRAMMBELEGUNGSADRESSE SETZEN

STACKSIZE OF "FILENAME" TO "NN"
ANWENDERSTACK-TIEFE SETZEN

```
SET BYTE_COUNT OF "FILENAME" TO "NN"
                ANZAHL DER BYTES IM LETZTEN RECORD

SET ENTRY_POINT OF "FILENAME" TO "NN"
                PROGRAMMEINTRITTSPUNKT FUER PROCEDUREFILES

ECHO ON .OR. OFF
                BETRIEBSART DER CONSOLE MIT ODER OHNE ECHO

AUTOLF ON .OR. OFF
                AUTOMATISCHES EINFUEGEN VON ZEILENVORSCHUEBEN
                FUER DIE KONSOLE ZU- BZW. ABSCHALTEN

DISKCON= "T0" "T1" "T2" "T3"
                BESTIMMT DIE LAUFWERKSKONFIGURATION FUER LAUF-
                WERK 0...3 ENTSPRECHEND T0...T3.
                DABEI GILT T#=0: LAUFWERK NICHT VORHANDEN
                           =2: AUFZ.PRINZIP MFM(5,25"),77 SP.
                           =5:      "      MFM(5,25"),40 SP.
                           =8:             FM(8"). 77 SP.

     %SET LINDEL=! CHRDEL=@

NACH DEM KOMMANDO KOENNEN ZEILEN MIT DEM ZEICHEN '!' UND
EINZELNE ZEICHEN MIT DEM SYMBOL @ GELOESCHT WERDEN.

        %SET DISKCON = 8 8 5 5

  BEI EINEM RECHNER MIT 4 FLOPPY-DISK-LAUFWERKEN SIND DIE
  LAUFWERKE 0 UND 1  8"-LAUFWERKE UND DIE LAUFWERKE 2 UND 3
  MINIFLOPPY-LAUFWERKE (40 SPUREN).
  ACHTUNG: IN MISCHKONFIGURATIONEN (SOWOHL STANDARD- ALS
  AUCH MINIFLOPPIES) IST DIE DISKETTENKONFIGURATION
  I M M E R  UEBER DAS SET-KOMMANDO ANZUGEBEN.
```

7.2.35. STATUS [0 .OR. 1 .OR. 2 .OR. 3]

```
AUSGABE DER DISKETTENSTATISTIK DES ANGEGEBENEN LAUFWERKES (AUF
UNIT 2).
DIE STATISTIK GIBT DIE ANZAHL DER BELEGTEN UND DER FREIEN
SECTOREN AUF DER DISKETTE AN.

OHNE ANGABE DER LAUFWERKSNUMMER:
        ANGABE ALLER STATISTIKEN ALLER ACTIVEN LAUFWERKE.

STIMMT DIE SUMME DER FREIEN UND DER BELEGTEN SEKTOREN NICHT MIT
GESAMT-SEKTORENZAHL UEBEREIN, ERFOLGT DIE MELDUNG:
        DISK STATISTICS ARE INCONSISTENT

WENN DIE ANZAHL DER ALS NICHT BELEGT MARKIERTEN SECTOREN UND
DIE ANZAHL DER WIRKLICH FREIEN SECTOREN NICHT UEBEREINSTIMMEN,
ERFOLGT DIE MELDUNG:
        WARNING: ALLOCATION IS INCONSISTENT.

DIESE FEHLER SIND FEHLER IN DER FORMATIERUNG DER DISKETTE.
ES IST DENNOCH MOEGLICH, ALLE FILES OHNE DATENVERLUST VON DER
DISKETTE ZU LESEN.
(DANN NEU FORMATIEREN!!)

        %STATUS 0
         DRIVE 0          DISKETTE MEIER
         256 SECTORS USED
         784 SECTORS AVAILABLE
```

7.2.36. VERBOSE

UEBERGANG IN VERBOSE-BETRIEBSART ('LANGE BETRIEBSART')
ALLE KOMMANDOKETTEN WERDEN ALS ECHO AUF DER KONSOLE WIEDERHOLT.

7.2.37. XEQ [* .OR. NN[PARAMETERLISTE]]

- WENN * UND PARAMETERLISTE ANGEGEBEN WAR: DAS ZULETZT GE-
  LADENE PROGRAMM WIRD UNTER VERWENDUNG DER OPTIONALEN
  PARAMETERLISTE GESTARTET.
- WENN NN UND OPT. PARAMETERLISTE ANGEGEBEN IST:
  PROGRAMMSTART AUF ADRESSE NN (HEXADEZIMALE ADRESSE).

```
%X 5600 P1 P2
```

SPRUNG ZUR ADRESSE 5600H, DER ZEILENPUFFERZEIGER (INPTR)
ZEIGT AUF DAS TRENNZEICHEN HINTER '5600' IN DER KOMMANDO-.
ZEILE

7.2.38. : "AUSDRUCK"

BERECHNET HEXADEZIMALE AUSDRUECKE VON LINKS NACH RECHTS UND
GIBT DAS RESULTAT AUS. ES DUERFEN DIE OPERATOREN +, -, * , /
VERWENDET WERDEN. UEBERLAUF WIRD NICHT BERUECKSICHTIGT.
KLAMMERN SIND UNZULAESSIG.

```
%: FD00-4400/80
0172
```

7.2.39. HELP ASCII ['ASCIIZEICHEN'*]

DER HEXADEZIMALE ASCII-CODE DES ZEICHENS WIRD AUSGEGEBEN.
DER SCHRAEGSTRICH (V.L.O.NACH R.U.) WIRD BENUTZT, UM SONDER-
ZEICHEN EINZUGEBEN. WIRD * ALS ARGUMENT VERWENDET, WIRD DIE
ASCII-CODE-TABELLE AUSGEGEBEN.
OHNE ARGUMENT: BESCHREIBUNG DES HELP ASCII KOMMANDOS.

```
%HELP ASCII 'A
ASCII   HEX
 'A'    41H
```

7.2.40. CHAR HEXZAHL[.OR. *]

OHNE OPTION - BESCHREIBUNG DES HELP CHAR KOMMANDOS, MIT * AUS-
GABE DER ASCII-CODELISTE. BEI ANGABE EINER HEXZAHL, MIT ODER OHNE
NACHGESTELLTEN H WIRD DAS ZUGEHOERIGE ASCIIZEICHEN AUSGEGEBEN.

```
%HELP CHAR 42
ASCII   HEX
 'B'    42H
```

7.2.41. (ERROR .OR. HELP ERROR) [ERRORCODE .OR. *]

AUSGABE DER BEDEUTUNG EINES FEHLERCODES, DER ALS 'COMPLETION
CODE' NACH RUECKKEHR INS BETRIEBSSYSTEM ODER NACH RUECKKEHR
AUS EINEM DEVICE (GERAETEPROGRAMM) IM PARAMETERVECTOR STEHT.
* ALS ARGUMENT: AUSGABE DER TABELLE DER ERROR-CODES.

8. Das UDOS-Dienstsystem

8.1. Dateiverwaltung Floppy-Disk

Der Zugriff auf die Daten einer Diskette bedingt eine eindeutige Struktur. Wie unter Punkt 3. und 8.2.2. beschrieben, werden die Daten in Sätzen zusammengefaßt, die 1,2,4,8 oder 16 Sektoren lang sein können. Solche logisch zusammengehörigen Sätze bilden die Datei. Sie kann z.B. ein Programm oder einen Text beinhalten. Die nähere Kennzeichnung der Datei erfolgt durch Attribute und Merkmale, die im DIRECTORY und DESCRIPTOR (Punkt 8.2.3.1. und 8.2.3.2.)ihren Niederschlag finden. So wird eine Datei durch ihren Namen, Dateityp, einer Reihe besonderer Merkmale und die Satzlänge spezifiziert.

8.1.1. Dateiname, Dateityp, Merkmale

Der Name einer Datei kann maximal 32 ASCII-Zeichen betragen. Das erste Zeichen muß ein Buchstabe sein. Wird der Name mit Punkt "." getrennt, so dient die Angabe hinter dem Punkt der Namenserweiterung. Das kann z.B. eine vom Anwender festgelegte Versionsnummer sein.
Weiterhin erkennt man aus der Angabe nach dem Punkt, ob es sich z.B. bei gleichen Dateinamen um Quelle, Objekttext oder Phase handelt (BSP.: Dateiname.S; definiert eine Quelle).

Der Dateityp wird vom Anwender beim Erstellen festgelegt. Es sind folgende Dateitypen möglich:

D	DIRECTORY	Diskenteninhaltsverzeichnis (s.8.2.3.) (wird automatisch beim Formatieren der Diskette generiert, ist nur einmal pro Diskette vorhanden)
A	ASCII	Datei besteht aus Text im ASCII-Code (BSP. editierte Programme)
B	BINARY	Datei mit binären Informationen (BSP. assemblierter Modul)
P	PROCEDURE	Dateien bzw. Programme, die sofort ladbar und abarbeitbar sind

Im Subtyp kann durch Angabe einer Ziffer zwischen Ø und 15. zwischen Dateien eines bestimmten Typs unterschieden werden.
Folgende Merkmale (Eigenschaften) können in Variation zugewiesen werden:

E	ERASE PROTECTED	löschgeschützt
L	LOCKED ATTRIBUTES	geschützt gegen Änderung dieser Eigenschaften
R	RANDOM	Datei mit wahlfreiem Zugriff
F	FORCE	Datei wird unabhängig von der aktuellen Hauptspeicherbelegung geladen (entspricht internem FORCE-Kommando)
S	SECRET	geheime Datei, wird bei normalen Aufrufen nicht gefunden
W	WRITE PROTECTED	Datei kann nicht beschrieben werden (schreibgeschützt)
*		Datei ohne spezielle Eigenschaften

Ein Aufruf der Datei zur Abarbeitung. erfolgt durch Angabe ihres Namens. Im allgemeinen gliedert man UDOS-Dateinamen in drei Teile:

1. - Name des Dateiorganisations- und Zuordnungsprogrammes (I/O-Treiber), beginnt immer mit dem Zeichen "¤"
   - abgeschlossen mit ":"
2. - Laufwerksnummer
   - abgeschlossen mit "/"
3. - eigentlicher Dateiname
   - [[NAME I/O-TREIBER:] [Laufwerk/]] NAME DATEI

z.B.: ¤ ZDOS:Ø/STATUS

Fehlt der Treiberprogrammname, verwendet das System den festgelegten MASTER. Nach Initialisierung von UDOS ist dies grundsätzlich ZDOS.
Eine Änderung der Zuweisung ist durch das MASTER-Kommando möglich. Ist im Programmnamen der Treibername und das Laufwerk oder nur "/" angegeben, wird die Datei als ein

externes Kommando betrachtet, ohne Angabe wird das Kommando intern bewertet. Treiberprogrammnamen sind im Prinzip Namen mit vorangestellten " ✖ Zeichen".
"✖" ist nicht mit Bestandteil des Namens.
Im System UDOS werden die Laufwerksnummern Ø bis 3 behandelt. Dabei ist die Anzahl der tatsächlich angeschlossenen Laufwerke zu beachten. Bei der Angabe ✱ werden alle Laufwerke durchsucht.

8.1.2. Floppy-Disk Laufwerkszuweisung

Die Laufwerke haben eine Zuordnung von Ø bis 3.
Bei der Arbeit mit UDOS-Kommandos zur Dateimanipulation ist u.a. durch Angabe der Laufwerksnummer eine Änderung der Standardzuweisung möglich. Eine variable Arbeit bei Anschluß mehrerer Laufwerke ist möglich.

8.1.3. UDOS-File-Debugger

Allgemeine Beschreibung

Das FILE.DEBUG Systemprogramm bietet dem Anwender Unterstützung beim Wiederaufbau zerstörter Floppy-Disk-Dateien. Mit seiner Hilfe können die folgenden Funktionen ausgeführt werden:

-Direktes Lesen und Schreiben von Sätzen auf der Diskette
-Lokalisieren des Dateinamens im Inhaltsverzeichnis der Diskette und Überprüfung des Descriptors der Datei
-Suchlauf durch eine Datei oder eine Adreßfolge,bis ein Zeiger mit dem Inhalt OFFFFH, d.h. Dateiende gefunden ist
-Identifikation aller Floppy-Disk-Sektoren nach gegebenem Inhalt
-Nutzung des DEBUG-Kommandos zur Korrektur des Pufferinhaltes vor Schreiboperationen

WARNUNG:

> Bei nicht sachgemäßem Gebrauch des Programmsystems FILE.DEBUG ist die Zerstörung einzelner Dateien oder der gesamten Diskette möglich.

Das Programmsystem FILE.DEBUG stützt sich auf den nachfolgend aufgeführten Satz von Variablen:

Variable	Beschreibung
DA	Aktuelle Diskadresse; sie wird bei allen Leseoperationen aktualisiert. Alle Diskadressen

enthalten zwei Byte:
- 1.Byte: Spuradresse
- 2.Byte: Laufwerk/Sektoradresse

Die Laufwerkadresse ist in den bit's 7-5, die Sektoradresse in den bit's 4-0 enthalten. Der durch DA adressierte Sektor wird mit '(DA)' gekennzeichnet.

BACK — Rückwärtszeiger von (DA); er wird bei allen Leseoperationen und optional auch bei Schreiboperationen aktualisiert.

FORE — Vorwärtszeiger von (DA); er wird bei allen Leseoperationen und optional auch bei Schreiboperationen aktualisiert.

RL — Record-Länge; sie wird gesetzt durch die Record-Länge der eröffneten Datei, oder wird optional bei Lese- und Schreiboperationen benutzt.

Die durch FILE.DEBUG ausgelisteten Adressen werden ohne Laufwerksangabe (bit's 7-5 des zweiten Byte=Ø) ausgegeben. Eine Laufwerkangabe innerhalb einer Adressangabe, als Teil eines Kommandos (bei READ oder WRITE) oder als Teil des Dateinamens (bei TRACE oder OPEN) veranlaßt, daß dieses Laufwerk auch bei den folgenden Operationen benutzt wird. Lese- oder Schreiboperationen,die auf andere Laufworke als Ø zugreifen,müssen die Laufwerksadresse in der Diskadresse enthalten.

FILE.DEBUG Kommandos

READ (Diskadresse(Record-Länge))
Liest Sektoren mit der angegebenen Satzlänge (impl.=128) beginnend bei FORE (oder einer angegebenen Diskadresse) in den Puffer (Startadresse=Linkadresse+CØØH)

Ausgabeformat:

DA = nnnn RL = nnnn BACK =nnnn FORE = nnnn

(CR) READ Kommando ohne Parameter, beginnend bei FORE.

(^) READ Kommando ohne Parameter mit umgekehrter Richtung, d.h. beginnend bei BACK.

WRITE (Diskadresse (Record-Länge (Rückwärts-Zeiger (Vorwärts-Zeiger))))

Schreibt Sektoren mit der angegebenen Record-Länge (impl. = 128) beginnend bei (DA) oder einer angegebenen Diskadresse vom Puffer (Startadresse = Linkadresse+ CØØH). Bevor die Schreiboperation beginnt, erfolgt nochmals die Abfrage:

DA =nnnn RL = nnnn BACK = nnnn FORE = nnnn ?

Ein 'Y' startet die Schreiboperation, ein anderes Zeichen führt zum Abbruch der Kommandoausführung.

OPEN Dateiname
Im Inhaltsverzeichnis wird der angegebene Dateiname gesucht und der Descriptor der Datei eingelesen.
Der Dateiname kann die Laufwerkadresse enthalten (impl. = Ø)
Jede Leseoperation während des Suchens im Inhaltsverzeichnis verursacht eine unter dem READ-Kommando erläuterte Ausgabe.
Wenn die angegebene Datei gefunden ist, erfolgt eine zusätzliche Ausgabe in der Form:

DSA = nnnn FRA = nnnn LRA = nnnn

Directory Sector Address	First Record Address	Last Record Address

Der Descriptor wird in den Puffer gelesen (Startadresse = Linkadresse + CØØH)

TRACE (Dateiname/^)
Es werden fortlaufend Leseoperationen durchgeführt, bis ein Vorwärts- oder Rückwärtszeiger Null ist oder ein Verkettungsfehler gefunden wurde. Wird ein Dateiname angegeben, so wird vor dem ersten Lesen zusätzlich eine OPEN Operation ausgeführt. Das "T" oder "T Dateiname"-Kommando führt den Suchlauf

vorwärts aus; das "T^" Kommando führt ihn rückwärts aus.

LIST — Die aktuellen Werte von DA, RL, BACK und FORE werden in der unter READ erläuterten Form ausgegeben.

DEBUG — Aufruf des Debuggers. Durch Eingabe eines "Q"-Kommandos wird der Debugger wieder verlassen, es erfolgt die Rückkehr zum FILE.DEBUG

QUIT — Verlassen des FILE.DEBUG Systems und Rückkehr zum UDOS.

SEARCH (Diskadresse)
Auf dem aktuellen Laufwerk werden Sektoren gesucht, deren Inhalt - nach einer Maskierung mit dem MASK-Puffer (Linkadresse + BØØH) - mit dem Inhalt des MATCH-Puffers (Linkadresse + B8ØH) übereinstimmt. Die Puffer werden während der Initialisierung = Ø gesetzt.
Setzt man im MASK-Puffer ein Bit (d.h. = 1), so wird dieses Bit im gelesenen Sektor mit dem entsprechenden Bit im MATCH-Puffer verglichen. Alle Diskadressen, bei denen der Vergleich positiv ausfällt, werden ausgegeben.

Wird eine Diskadresse angegeben, so ist das Vergleichskriterium nicht der Sektorinhalt. Es wird geprüft, ob die angegebene Diskadresse mit einem Vorwärts- oder Rückwärtszeiger der gelesenen Sektoren übereinstimmt. Es wird nicht angezeigt, bei welchem Zeiger die Übereinstimmung auftrat.
Die SEARCH Operation kann durch Eingabe von ET2 abgebrochen werden.

8.2. ZDOS

8.2.1. Allgemeines

ZDOS ist das Gerätesteuerprogramm für Floppy-Disk-Arbeit. Es wird vom Systemkern als " Master-Device", d.h. zum Laden und Ablegen von Programmen und Daten genutzt. Es kann aber auch durch Anwenderprogramme aufgerufen werden.

Die Aufruf- und Rückkehrparameter entsprechen der universellen I-O-Vektorschnittstelle.
Die Grundstruktur der Daten auf der Diskette ist die Datei, die in physischen Sätzen mit Längen von 128, 256, 512, 1024, 2048 Byte abgelegt werden kann. Der Zugriff erfolgt über den Dateinamen, dem als erstes eine logische Nummer zugeordnet wird. Im ZDOS kann gleichzeitig mit 16 eröffneten Dateien (log. Nummern) gearbeitet werden.
Die Namen aller Dateien werden im Verzeichnis, das selbst eine Datei ist, mit dem Namen "Directory" festgehalten. Für die Arbeit mit Zwischendaten bildet die "SCRATCH" - Datei eine Ausnahme.
Wird einer "logischen Nummer" eine Datei ohne Namen (Dateinamenlänge=Ø) zugewiesen, so wird eine Registrierung im Directory unterdrückt, so daß nach Schließen der Datei auf sie nicht zugegriffen werden kann.

8.2.2. Diskettenbelegung

Um eine maximale Auslastung der Disketten und eine geringe Zugriffszeit zu gewährleisten, werden die Daten und die Sätze der Dateien nicht physisch hintereinander abgespeichert. Jeder physische Satz (Record) hat zwei Zeiger, die den logisch davor-und dahinterliegenden Satz adressieren und damit eine Satzkettung für die gesamte Datei vornehmen. Dadurch ist es möglich, nach Streichen einer Datei die frei gewordenen Sektoren für andere Dateien zu verwenden. Nach Möglichkeit wird die Satzdistanz für eine optimale Zugriffszeit eingestellt.
Diese Tätigkeiten werden ebenfalls vom ZDOS verwaltet. Dazu existiert auf jeder Diskette ein Belegungsplan (Spur 17H, Sektor Ø...2), der neben dem Diskettennamen für jeden Sektor der Diskette ein Bit bereithält. Dieses Bit signalisiert die Belegung des entsprechenden Sektors durch eine Datei und wird demzufolge durch jeden Schreib- bzw. Löschvorgang aktualisiert.
Damit beim Wechsel von Disketten keine unerwünschte Fehlbelegung auftritt, muß man deshalb im OS das Kommando "I" ausführen.
Dieses Kommando bewirkt für das Floppy-System das Einlesen der Belegungspläne in den Hauptspeicher.

8.2.3. Satzaufbau und Dateiorganisation

Die Disketten-Dateistruktur ist so aufgebaut, daß die Daten in einem oder mehreren Sätzen zusammengefaßt werden. Eine Datei besteht aus einem "Descriptor-Satz" (Dateibeschreibung mit Satzlänge 128) und ein oder

mehreren Datensätzen gleicher Länge (128...2048 Byte).
Jeder Satz enthält zwei durch ZDOS erzeugte Zeiger,
die zum vorangehenden bzw. nachfolgenden Satz zeigen.
Die Datei ist also eine Menge miteinander verketteter
Sätze. Dateien werden mit ihren Namen aufgerufen.
Das Verzeichnis der Dateien befindet sich auf der Spur 16H
als DIRECTORY. Das Directory ist selbst eine Datei.
Für jede Datei gibt es im Verzeichnis eine Eintragung, die
den Dateinamen (1Byte Länge des Dateinamens und 1...32Byte
Namen) und den Zeiger zum Descriptor der Datei enthält.

8.2.3.1. Aufbau DESCRIPTOR

Der Descriptor-Satz ist der erste Satz einer Datei.
Auf diesen Satz zeigt die jeweilige Adresse im Directory.
Der Satz gehört mengenmäßig und inhaltlich nicht zu den
Sätzen der Datei. Er beinhaltet zusätzlich Informationen
über die Dateien.
Der Descriptor-Satz besteht immer aus 1Sektor zu 128 Byte,
unabhängig von der Satzlänge der Datei. Von den 128 Byte
werden durch ZDOS 40 Byte definiert, die restlichen können
vom Anwender belegt werden.

Byte	Inhalt
0 - 3	reserviert
4 - 5	nicht verwendet
6 - 7	Zeiger zum DIRECTORY-Sektor mit zugehöriger Datei- Information
8 - 9	Zeiger zum ersten Datensatz der Datei
10 - 11	Zeiger zum letzten Datensatz der Datei
12	Datei-Typ (siehe Open-Aufruf)
13 - 14	Anzahl der Sätze der Datei
15 - 16	Satzlänge
17 - 18	gleich der Satzlänge gesetzt
19	Schutzeigenschaften der Datei (siehe Open-Aufruf)
20 - 21	Startadresse (nur bei Prozeduren)

22 - 23	reserviert
24 - 31	Dateierstellungsdatum
32 - 39	Datum der letzten Änderung
40 - 127	von ZDOS nicht benutzt

Außer den Bytes Ø-11 ist der Inhalt des Descriptors bei folgenden Aktionen verfügbar:

Öffnen der Datei	(sh. OPEN)
Öffnen der Datei bei Anfrage nach den Merkmalen	(sh. QUERY-ATTRIBUTES)
Erstellung der Datei	('Creation of the File')
Änderung der Merkmale	(sh. SET-ATTRIBUTES)
Datei auf neuesten Stand bringen	(sh. UPDATE)
Schließen der Datei	(sh. CLOSE)

8.2.3.2. Datei-Zeiger

Beim Zugriff auf eine Datei sind drei Zeiger zu aktivieren:

- Zeiger auf den laufenden Satz
- Zeiger auf den vorhergehenden Satz
- Zeiger auf den Folgesatz

Beim Öffnen einer Datei wird zunächst im DIRECTORY gesucht, danach der dazugehörige DESCRIPTOR-Satz gelesen und anschließend werden die drei Zeiger initialisiert. Von diesem Zeitpunkt an gilt eine Datei als aktiv.

8.2.4. Aufrufkonventionen und Kommandoparameter

Das Betriebssystem bearbeitet alle über UDOS durchgeführten Ein- und Ausgabeaufrufe (I/O-Requests), indem es das zugewiesene Treiberprogramm und die logischen Einheiten aktiviert. Dazu benötigt das Betriebssystem Parameter, die vom Anwenderprogramm zur Verfügung zu stellen sind. Im Register IY ist die Adresse einzutragen, ab der die Parameterbytes eingetragen sind.

Es gibt zwei Arten des I/O-Requests:

- die gesamte I/O-Operation wird vom Treiber vollständig abgearbeitet und anschließend an das aufrufende Programm zurückgekehrt (RETURN ON COMPLETION)

- es wird nur der Start der I/O-Operation vom Treiber durchgeführt, danach wird zum aufrufenden Programm zurückgekehrt und der Treiber arbeitet über Unterbrechung weiter (IMMEDIATE RETURN)

Bei Rückkehr ins aufrufende Programm wird im 11.Parameterbyte der sogenannte Fertigstellungscode (Completionscode) zurückgegeben, den das aufrufende Programm auswerten kann.

Kommandoparameter

Die Adresse in IY zeigt auf das 1.Byte der Parameter.

Byte:	1	Logische Nummer Es können 256 logische Funktionen (Ø...FFH) benutzt werden, wovon aber gleichzeitig nur 16 aktiv sein dürfen.
	2	Operationscode Er identifiziert die gewünschte Operation. Soll eine Rückkehr nach Fertigstellung der Operation erfolgen,ist Bit Ø=Ø. Wird BitØ mit 1 belegt, erfolgt Interrupt-Abarbeitung.
	3,4	Datenbereichsanfang Wenn keine Datenübertragung erforderlich, ist Adresse mit ØØØØ zu belegen.
	5,6	Länge des Datenblockes Bei Rückkehr ist hier die Länge der tatsächlich übertragenen Bytes eingetragen.
	7,8	Rückkehradresse Ist bei Unterbrecherarbeit zu belegen.
	9,10	Fehlerrückkehradresse Bei Ø wird sie übertragen, bei ≠Ø wirkt die Eintragung als Fehlerrücksprungadresse, wenn ein Fehler bei Abarbeitung des Requests auftrat.

11 Fertigstellungscode
Byte 11 wird vor Aufruf auf Ø gesetzt, nach der Rückkehr bedeutet

Bit 7 = 1 kein Fehler sonst
Bit 6 = 1 Fehler

Die Fehlerart wird durch die restlichen Bits gekennzeichnet.

12,13 Adresse, die auf Zusatzparameter zeigt oder 2 Bytes Zusatzinformationen

8.2.5. Rückmeldungen

8.2.5.1. Allgemeine Meldungen

Bei diesen Meldungen ist das Bit6 nicht gesetzt.

ØØ Rückmeldung bei nicht vollständig abgearbeitetem Request im Unterbrechungsmode

80 Request wurde vollständig und **fehlerfrei** abgearbeitet

81 Formatfehler im Directory

82 es wurde eine Scratch-Datei erzeugt

83 Dateiname ist zu lang, Rest wird abgeschnitten

84 die Merkmalsliste ist zu lang, wird abgeschnitten

benutzte Register

ZDOS benutzt den Zweitregistersatz (A'.....L') und das IY-Register.

8.2.5.2. Fehlermeldungen

Eine Rückmeldung bei Fehlern erfolgt über das Byte 11 der Kommandoparameter. Der Fehlercode ist hexadezimal angegeben.

Fehlercode	Bedeutung
C1	keine gültige Operation
C2	keine Bereitschaft
C3	schreib- bzw. löschgeschützt
C4	Sektoradreßfehler
C5	Suchfehler
C6	Datenübertragungsfehler (CRC-Fehler)
C7	Datei ist nicht vorhanden
C8	-
C9	Fehler EOF
CA	Fehler bei Zeigerkontrolle
CB	Datei nicht geöffnet
CC	"logische Einheit" bereits aktiv
CD	Zuweisungsspeicherpuffer gefüllt
CE	ungültiges Diskettenlaufwerk
CF	Tabelle der "logischen Funktionen" bereits gefüllt (bereits 16 logische Funktionen aktiv)
DØ	Datei bereits vorhanden
D1	Identifizierungsfehler Diskette
D2	Eigenschaften ungültig
D3	Diskette voll
D4	DIRECTORY enthält Datei nicht
D5	Dateianfang falsch
D6	Datei ist bereits geöffnet
D7	Umbenennung ist ungültig
D8	Datei ist gesperrt (bei Versuch einer Eigenschaftsänderung)
D9	ungültiger OPEN-Aufruf

8.2.6. Aufrufe

8.2.6.1. INITIALIZE

Vektor:

Logische Funktion	- nicht verwendet
Code	- ØØ oder Ø1
Datenstartadresse	- nicht verwendet
Datenblocklänge	- nicht verwendet, es wird ØØ zurückgemeldet
Rückkehradresse	
Fehler-Rückkehradresse	
Folge-Vektor	- keiner

Aktion: Alle 16 „logischen Funktionen" werden als nicht offen markiert. Die Tabelle der „logischen Funktionen" wird gelöscht. Der gesamte Anwenderspeicher wird wieder freigegeben (s. auch RELEASE-Kommando). Die Diskettenbelegungspläne werden eingelesen und eine Anzeige gesetzt, welche über die aktiven Laufwerke Auskunft gibt.

Mögliche Fehler:

C4, C5, C6.	Der Aufruf wird bei Auftreten eines dieser Fehler nicht fertig durchgeführt.
DA.	Nicht ausreichender Platz für die Belegungspläne.

8.2.6.2. ASSIGN

Vektor:

logische Funktion	
Code	- Ø2 oder Ø3
Datenstartadresse	- nicht verwendet
Datenblocklänge	- nicht verwendet
Rückkehradresse	
Fehler-Rückkehradresse	
Folge-Vektor	- Zeiger zu weiteren Vektoren mit folgendem Inhalt:

Byte1	von ZDOS nicht verwendet
Byte2	Codezeichen des verwendeten Laufwerkes (Ø...3 oder *)
Byte3	Bytezahl im Dateinamen, max.32. Falls Ø ⟶ SCRATCH-Datei
Byte4,..	Dateiname

Aktion: Der angegebene Dateiname wird mit einer "log. Funktion" für nachfolgende I/O-Operationen verknüpft.
Vorangehende Zuweisungen werden gelöscht, der Prozedurname wird in einen Pufferspeicher für den späteren Gebrauch abgelegt (bei Aufruf der "log. Funktion").
Falls die Prozedur in der Zwischenzeit einmal eröffnet und wieder geschlossen wird, geht die Verbindung mit der "log. Funktion" verloren, der Aufruf ASSIGN muß wiederholt werden.

Mögliche Fehler:

CC. Versuch der Verknüpfung mit einer aktiven "log. Funktion"
CE. Ungültige Diskettenlaufwerks-Nummer (weder 'Ø...3', noch '*')
CD. Dateinamenpuffer voll
83. Dateiname war zu lang, wurde abgeschnitten

8.2.6.3. OPEN

Vektor:

logische Funktion
Code - Ø4 oder Ø5
Datenstartadresse - Zeiger zu Bereich mit Inhalt der Merkmale für erstellende Datei, bzw. mit Merkmalen für bereits bestehende Datei. Falls Ø, werden die üblichen Merkmale der zu erstellenden Datei übernommen
Datenblocklänge - Anzahl der Bytes für gespeicherte Merkmale. Falls Ø oder zu klein für die üblichen Merkmale, nicht verwendet.
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge-Vektor - Zeiger zu einem Bereich mit folgendem Inhalt:

Byte1 Information über durchzuführende Aktion
Byte2 Kennzeichen des Diskettenlaufwerkes ('Ø'...'3') auf der die Aktion durchgeführt wurde

Byte3 Länge des Dateinamens.
Der OPEN-Aufruf tätigt seine eigenen Zuweisungen (ASSIGN), wodurch nicht zwei Aufrufe von ZDOS nötig sind. Die benötigten Informationen müssen denen des ASSIGN-Aufrufs entsprechen. Falls kein ASSIGN stattfinden soll, muß dieses Byte / ØFFH / gesetzt sein.

Byte4 Dateiname, falls vorhanden

Datei-Merkmale:

Jede Datei enthält einen speziellen Satz (mit 128 Bytes), den Descriptor-Satz (sh. 8.2.3.1.). Bei der Erstellung einer Datei muß ihre Organisation vom Anwender entworfen werden, ansonsten erfolgt sie in üblicher Weise. Die Öffnung einer Datei durch ein Programm erfordert die Übergabe der Informationen über die Dateiorganisation an das öffnende Programm.

Organisation des DESCRIPTORS:

Byte1 Typ und Subtyp . Durch Setzen eines der 4 oberen Bits ist folgende Spezifikation möglich:

Bit 7 - PROZEDUR
Bit 6 - DIRECTORY
Bit 5 - ASCII
Bit 4 - BINARY
Bit 3-Ø - können vom Anwender definiert werden. Ohne Angabe wird ASCII Subtyp 0 (2ØH) zugewiesen.

Bytes 2-3 Anzahl der Sätze der Datei

Bytes 4-5 Satzlänge in Bytes. Standardzuweisung 128 Bytes, wenn Bytes 4 und 5 gleich Null sind

Bytes 6-7 Blocklänge in Bytes , derzeit gleich Satzlänge.

Byte8 Schutzeigenschaften

Bit 7 - Schreibgeschützt
Bit 6 - Löschgeschützt
Bit 5 - Blockiert, keine Eigenschaft ist änderbar

Bit 4 - Geheim (bei normalen DIRECTORY-Ausdruck)
Bit 3 - Wahlfrei, die Datei kann mit wahlfreiem Zugriff ausgelesen werden.
Bit 2 - Datei wird unabhängig von der aktuellen Hauptspeicherbelegung geladen.
Bit 1, Bit Ø - Für Systemgebrauch reserviert.

Bytes 9- 10 Startadresse einer Prozedur. Von ZDOS selbst nicht verwendet.

Bytes 11- 12 Für Systemgebrauch reserviert.

Bytes 13- 20 Datum der Erstellung.

Bytes 21- 28 Datum der letzten Modifikation.

Bytes 29-116 Frei für Anwenderdefinition.

Varianten des OPEN-Aufrufs:

Die verschiedenen Arten der Aktivierung einer Datei werden durch das 1.Byte des Folge-Vektors spezifiziert. Eine Datei kann für folgende Zugriffe eröffnet werden:

a) Wahlfreier Zugriff (RANDOM ACCESS), spezifiziert durch Setzen von Bit3 dieses Bytes, hier ergeben sich zwei Folgerungen: der READ DIRECT-Zugriff wird angenommen (ansonsten INVALID REQUEST-Fehler) und bei jeder satzorientierten Aktion wird die Adresse des ersten angesprochenen Satzes an das aufrufende Programm zurückgeliefert (im Folge-Vektor des Parameter-Vektors)

b) Sequentieller Zugriff.

Es existieren fünf einander gegenseitig ausschließende Verfahren für die Handhabung der/'DATEI GEFUNDEN' / 'DATEI NICHT GEFUNDEN' /-Fälle.
Sie werden in den unteren 3 Bits des 1. Bytes des Folge-Vektors angegeben:

OPEN for INPUT Code Ø

Sofern die Datei existiert, wird sie aktiviert, wobei der Zeiger auf den Beginn des 1. Satzes weist; bei Nichtexistenz erfolgt die Fehlermeldung 'FILE NOT FOUND' (Code C7).

OPEN for OUTPUT Code 1

Sofern die Datei existiert, wird sie aktiviert und ihre sämtlichen Sätze werden gelöscht; bei Nichtexistenz wird sie erstellt.

OPEN NEW FILE Code 2

Öffnen für nichtzerstörende Ausgabe.
Sofern die Datei existiert, folgt die Fehlermeldung 'DUPLICATE FILE' (Code DØ) und die Datei wird nicht aktiviert; bei Nichtexistenz wird sie erstellt.

OPEN for APPEND Code 3

Sofern die Datei existiert, wird sie aktiviert und der Zeiger weist auf das Ende des letzten Satzes; bei Nichtexistenz wird sie erstellt.

OPEN for UPDATE Code 4

Sofern die Datei existiert, wird sie aktiviert und der Zeiger zeigt zum Beginn des ersten Satzes; bei Nichtexistenz wird sie erstellt.

Aktion: Sofern die Zuweisung eines Dateinamens gewünscht wurde (Byte 3 des Folge-Vektors ist nicht ØFFH), wird eine ASSIGN-Subroutine aufgerufen, als ob ein ASSIGN-Aufruf erfolgt wäre. In diesem Fall wird das DIRECTORY des aufgerufenen Diskettenlaufwerkes nach dem Dateinamen durchsucht (beim Code '*' erfolgt eine Überprüfung der READY-Information aller vorhandenen Laufwerke, bis die Datei gefunden wurde oder sämtliche möglichen Laufwerke durchsucht sind.

Die ID (Identifikation) jener Diskette, welche entweder die Datei enthält oder auf der die Datei erstellt wird, kommt in einen Pufferspeicher und wird mit der ID der korrespondierenden Liste im Speicher verglichen.

Wenn sie nicht übereinstimmen und keine andere Datei im Moment auf **demselben Diskettenlaufwerk** offen ist, wird eine neue Liste (und ID) angelegt und in den Speicher geschrieben. Falls eine Datei offen ist, erfolgt eine Fehlermeldung 'WRONG DISKETTE', die Datei wird nicht aktiviert. Beim Auffinden der Datei wird ihr DESCRIPTOR gelesen, seine wichtigen Teile in die ' Tabelle der aktiven Dateien' übernommen und sofern gewünscht, in den **Datentransferbereich** transferiert. Anschließend wird die Datei als 'OFFEN' markiert. Soll eine Datei erst erstellt werden, erzeugt man einen DESCRIPTOR-Satz in einem Pufferspeicher, transferiert ihn zur 'Tabelle der aktiven Dateien' und falls gewünscht, zum Datenspeicherbereich des Anwenders.
Hat die Datei einen Namen, wird sie auch zur Diskette geschrieben und erhält einen Platz im DIRECTORY.

Mögliche Fehler:

Grundsätzlich sind alle DISK ERRORS möglich.

NOT READY Code C2

Das angesprochene Diskettenlaufwerk war beim Aufruf nicht bereit.

PROTECTION Code C3

Die Diskette war schreibgeschützt, bzw. die aufgerufene Datei schreib- und löschgeschützt gegen 'OPEN for OUTPUT' (mit anschließendem Löschen ihrer Sätze). Im letzteren Fall wird die Datei eröffnet, aber nicht gelöscht.

UNIT ALREADY OPEN Code CC

Die 'log. Einheit' ist bereits aktiv. Sie muß entweder geschlossen oder neu initialisiert werden, um für OPEN bereit zu sein. In diesem Fall wird keine Aktion durchgeführt

WRONG DISKETTE Code D1

ID der Diskette im Laufwerk ist nicht identisch mit ID im Speicher. Das heißt, daß die Diskette nach der letzten INITIALIZE-Aktion ausgetauscht wurde, bzw. daß die Listen im Speicher von einem Programm überschrieben wurden. Die Datei wird nicht eröffnet.

FILE NOT FOUND Code C7

Der OPEN-Aufruf war für eine Eingabe gedacht und die aufgerufene Datei existierte nicht.

POINTER ERROR Code CA

Die Sektorenzeiger für die Verbindung der verschiedenen Teile sind falsch oder zerstört oder die Zeiger des DESCRIPTORS sind inkorrekt.

DUPLICATE FILE Code DO

Aufruf zum Öffnen einer neuen Datei, während die Datei bereits existiert. Die Datei wird nicht aktiviert.

INVALID ATTRIBUTE Code D2

Eine spezifizierte Eigenschaft war ungültig oder die Satzlänge ist inkorrekt. Die Datei wird mit den automatisch zugewiesenen Eigenschaften anstelle der inkorrekten aktiviert.

DISK FULL Code D3

Es ist kein Platz für einen DESCRIPTOR-Satz bzw. einen weiteren DESCRIPTOR-Satz verfügbar. Dieser Fehler kann nur bei der Dateierstellung auftreten.

FILE ALREADY OPEN Code D6

Die zu öffnende Datei ist bereits bei einer anderen 'logischen Funktion' aktiviert.

PROPERTIES PROTECTION Code D8

Versuch, die Eigenschaften bei einer blockierten Datei zu ändern

INVALID OPEN REQUEST Code D9

Ungültiger Aufruf, (weder INPUT, OUTPUT, NEWFILE, APPEND, UPDATE)

INSUFFICIENT MEMORY Code DA

Der Speicherplatz für die Belegungspläne der Disketten reicht nicht aus.

Warnungen:

- DIRECTORY FORMAT ERROR Code 81

  Das Format eines oder mehrerer DIRECTORY-Sätze war falsch. Der Satz kann gelesen werden, aber die Daten können falsch sein.

- SCRATCH FILE Code 82

  Eine SCRATCH-Datei wurde erstellt.

- ATTRIBUTES TOO LONG Code 84

  Mehr als 116 Bytes Merkmale.
  (nur 116 Bytes übertragen)

8.2.6.4. CLOSE

Vektor:

logische Funktion	
Code	-Ø6 oder Ø7
Datenstartadresse	-Zeiger zu einem Speicherbereich mit Eigenschaften, die die der Datei ersetzen. Format wie bei OPEN. Wenn eine Änderung der Eigenschaften unerwünscht ist, =Ø.
Datenblocklänge	- Anzahl der zum DESCRIPTOR zu transferierenden Bytes (max. 116), ansonsten =Ø
Rückkehradresse	
Fehler-Rückkehradresse	
Fertigstellungscode	
Folge-Vektor	-keiner

Aktion: Sofern die Datei eine SCRATCH-Datei war, wird sie gelöscht. Bei einer Änderung der Belegung oder der Eigenschaften der zu schließenden Datei, liest das System den DESCRIPTOR, bringt ihn auf den letzten Stand und ändert den Belegungsplan. Die Datei wird anschließend als geschlossen markiert.

Mögliche Fehler:

FILE NOT OPEN Code CB

Die 'logische Funktion' war nicht aktiv.

WRONG DISKETTE Code D1

Die Diskettenidentifikation ist nicht mit der im Speicher identisch. Das kann nach Austausch der Diskette ohne "INITIALIZE" oder Überschreiben der Belegungspläne durch ein Programm auftreten. Die Datei wird nicht abgeschlossen. Zur Behebung des Fehlers muß entweder die korrekte Diskette wieder eingesetzt werden. oder INITIALIZE ausgeführt werden.

INVALID ATTRIBUTE Code D2

Entsteht, falls der Dateityp nicht zu den Eigenschaften paßt. Alle DISK-ERRORS sind möglich.

POINTER ERROR Code CA

Zeigt einen Zeigerfehler an, der während des Löschens einer SCRATCH-Datei auftritt, bzw. daß die Zeiger eines DESCRIPTORS unrichtig sind. Falls der Fehler während des Lesens eines DESCRIPTORS auftritt, wird die Datei deaktiviert, der DESCRIPTOR bleibt jedoch unverändert.

Warnung: ATTRIBUTES TOO LONG Code 84

Es wurden mehr als 116 Bytes Eigenschaften angegeben, jedoch nur 116 Bytes transferiert (maximal).

8.2.6.5. REWIND

Vektor: logische Funktion
Code - Ø8 oder Ø9
Datenstartadresse - nicht verwendet
Datenblocklänge - nicht verwendet
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge-Vektor - keiner

Aktion: Der Datenzeiger wird zum Descriptor positioniert, wobei als nächster Satz der 1. Datensatz markiert wird. Diese Position wird beim Eröffnen der Datei (OPEN FILE) oder beim Erstellen (CREATING) eingenommen, nicht aber beim Erweitern (APPEND). Bei leerer Datei ist der Zeiger auf den nächsten Satz (NEXT RECORD POINTER) gleich Ø.

Mögliche Fehler:

FILE NOT OPEN Code CB

Die aufgerufene "logische Funktion" ist nicht aktiviert. Es wird keine Aktion durchgeführt.

8.2.6.6. READ BINARY

Vektor: logische Funktion
Code - ØA oder ØB
Datenstartadresse - Anfangsadresse, ab der die Daten gespeichert werden.
Datenblocklänge - Anzahl der zu lesenden Bytes. Falls diese Zahl kein Vielfaches der Satzlänge ist, wird sie aufgerundet. Enthält bei Rückkehr die Zahl der tatsächlichen transferierten Bytes.
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge-Vektor - Ist die Datei für wahlfreien Zugriff offen, enthält dieses Feld die Adresse eines Bereichs von drei Bytes, in welchen die Diskettenadresse des ersten gelesenen Satzes geschrieben wird.
Ansonsten nicht verwendet.

Aktion: Die Daten werden vom nächstfolgenden Satz der Datei gelesen und ab der Datenstartadresse abgespeichert. Der Zeiger zeigt auf den letzten gelesenen Satz.
Wurde die Datei für wahlfreien Zugriff eröffnet, enthält nach der Rückkehr der Folge-Vektor die Adresse des ersten gelesenen Satzes. Das dritte Byte dieser Adresse ist stets Ø.

Mögliche Fehler: Alle Disketten-Fehler außer PROTECTION möglich.

FILE NOT OPEN Code CB

Die aufgerufene "logische Funktion" ist nicht aktiviert. Es wird keine Aktion durchgeführt.

END OF FILE Code C9

Der letzte Satz der Datei wurde gelesen, ohne daß die Zahl der gewünschten Bytes erreicht wurde. Die rückgemeldete Datenblocklänge liefert eine Information über die tatsächliche Bytezahl.

8.2.6.7. WRITE BINARY

Vektor:

logische Funktion	
Code	- ØE oder ØF
Datenstartadresse	- Adresse, ab der die zu schreibenden Daten zu finden sind.
Datenblocklänge	- Die Zahl der zu schreibenden Bytes wird auf ein ganzzahliges Vielfaches der Satzlänge aufgerundet. Bei Rückkehr steht hier die Zahl der tatsächlich übertragenen Bytes.
Rückkehradresse	
Fehler-Rückkehradresse	
Fertigstellungscode	
Folge-Vektor	- Bei wahlfreiem Zugriff zeigt die hier enthaltene Adresse auf einen Drei-Byte-Bereich, der die Adresse des ersten beschriebenen Satzes enthält. ansonsten nicht verwendet.

Aktion: Es werden neue Sätze erzeugt und mit Daten aus dem Datenspeicher gefüllt. Die neuen Sätze werden hinter dem aktuellen eingeschoben. Der Zeiger weist auf den letzten eingeschriebenen Satz, der Zeiger für den nächsten Satz (NEXT RECORD POINTER) zeigt auf denselben Satz wie vor der Operation.

Mögliche Fehler: Alle Diskettenfehler außer CRC (Code C6). können auftreten.

PROTECTION Code C3

Meldung, falls die Datei schreibgeschützt war.

FILE NOT OPEN Code CB

Die aufgerufene "logische Funktion" war nicht aktiviert. Keine Aktion wird durchgeführt.

DISK FULL Code D3

Auf der Diskette ist kein Platz für weitere Daten, wobei die Diskette bereits mit einigen neuen Sätzen beschrieben worden sein kann. Die Datenblocklänge enthält die Anzahl der tatsächlich aufgezeichneten Bytes.

8.2.6.8. WRITE CURRENT

Vektor: logische Funktion
Code - 12 oder 13
Datenstartadresse - Anfangsadresse des Datenspeicherbereichs
Datenblocklänge - Falls = Ø, werden keine Daten transferiert, ansonsten 1 Satz. Bei der Rückkehr steht hier die Zahl der tatsächlich übertragenen Bytes.
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellunscode
Folge-Vektor - Bei Aufruf für wahlfreien Zugriff enthält dieses Feld die Adresse eines Drei-Byte-Bereichs mit der Diskettenadresse des Satzes; ansonsten nicht verwendet.

Es wird ein Satz vom Speicher zur Datei übertragen, welcher die Daten des momentanen markierten Satzes überschreibt. Dabei wird kein neuer Satz erzeugt, und der Satz-Zeiger (RECORD POINTER) bleibt unverändert.

Mögliche Fehler:

Alle Disketten-Fehler sind möglich, außer CRC (Code C6)

PROTECTION (Code C3) wird bei schreibgeschützten Dateien gemeldet.

FILE NOT OPEN Code CB
Die aufgerufene "logische Funktion" ist nicht aktiviert.

8.2.6.9. DELETE

Vektor: logische Funktion
Code - 16 oder 17
Datenstartadresse - nicht verwendet
Datenblocklänge - Anzahl der von der Datei gelöschten Bytes, wird auf einen vollen Satz aufgerundet. Enthältbei der Rückkehr die Zahl der gelöschten Bytes.
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge -Vektor - keiner

Aktion: Beginnend mit dem aktuellen Satz werden solange Sätze von der Datei gelöscht und ihr Platz freigegeben (DEALLOCATED), bis die gewünschte Anzahl von Bytes gelöscht wurde. Der Satz-Zeiger (CURRENT RECORD POINTER) bleibt auf dem letzten, den gelöschten Sätzen vorausgehenden Satz stehen.
Steht der Zeiger gerade auf dem DESCRIPTOR, wird er zum ersten Datensatz positioniert, bevor die Aktion abläuft. Anschließend zeigt er wieder auf den DESCRIPTOR.

Mögliche Fehler:

Alle Diskettenfehler einschließlich PROTECTION, falls die Datei schreib-oder löschgeschützt war, sind möglich.

END OF FILE Code C9

Der letzte Satz der Datei wurde gelöscht, ohne die gewünschte Zahl der zu löschenden Bytes zu erreichen. Die gemeldete Zahl von Bytes entspricht den tatsächlich gelöschten Bytes,

inclusive des letzten Satzes. Der Satz-Zeiger (CURRENT RECORD POINTER) zeigt auf den, dem ersten gelöschten Satz vorangehenden Satz; der Zeiger für den nächsten Satz (NEXT RECORD POINTER) ist Ø.

POINTER ERROR Code CA

Während des Löschens wurde ein Zeigerfehler entdeckt. Alle Sätze bis zum Auftreten des Fehlers werden gelöscht.

FILE NOT OPEN Code CB

Die aufgerufene "logische Funktion" ist nicht aktiv.

8.2.6.10. DELETE REMAINING RECORDS

Vektor:

logische Funktion	
Code	- 18 oder 19
Datenstartadresse	- nicht verwendet
Datenblocklänge	- nicht verwendet, rückgemeldet wird die Anzahl gelöschter Bytes.
Rückkehradresse	
Fehler-Rückkehradresse	
Fertigstellungscode	
Folge-Vektor	- keiner

Aktion: Alle Sätze einer Datei vom aktuellen bis zum letzten werden gelöscht. Der Zeiger bleibt auf dem Satz unmittelbar vor dem ersten gelöschten Satz stehen. Der Zeiger für den nächsten Satz ist Ø. Zeigt der Zeiger auf den DESCRIPTOR, wird er zuerst auf den ersten Datensatz positioniert.

Mögliche Fehler: Alle Diskettenfehler inclusive PROTECTION.

POINTER ERROR Code CA

Während des Löschens wurde ein Zeigerfehler entdeckt. Alle Sätze bis zum Auftreten des Fehlers werden gelöscht.

FILE NOT OPEN Code CB

Die aufgerufene "logische Funktion" ist nicht aktiviert.

8.2.6.11. ERASE

Vektor:

logische Funktion	
Code	- 1A oder 1B
Datenstartadresse	- nicht verwendet
Datenblocklänge	- nicht verwendet Die Zahl der gelöschten Bytes wird zurückgemeldet.
Rückkehradresse	
Fehler Rückkehradresse	
Fertigstellungscode	
Folge-Vektor	- keiner

Aktion:

Alle Sätze einer Datei einschließlich DESCRIPTOR werden freigegeben (DEALLOCATED). Der Einsprung im DIRECTORY wird gelöscht. Es muß kein OPEN-Aufruf erfolgen, jedoch ASSIGN ist nötig.

Mögliche Fehler: Alle Diskettenfehler sind möglich (inclusive PROTECTION).

NOT READY Code C2
Entsteht, falls das angegebene Laufwerk nicht aktiv ist.

FILE NOT FOUND Code C7
Die genannte Datei kann bei den spezifizierten Laufwerken nicht aufgefunden werden.

POINTER ERROR Code CA
Beim Aufruf bzw. Löschen der Sätze wurde ein Zeigerfehler entdeckt. Alle nachfolgenden Sätze bleiben als belegt markiert (ALLOCATED).

INVALID DRIVE Code CE
Das angegebene Laufwerk war ungleich Ø-3; "*" ist keine gültige Spezifikation.

WRONG DISKETTE Code D4
Der Disketten-ID entspricht nicht der Angabe im Speicher. Das kann beim Wechseln einer Diskette ohne nachfolgendem INITIALIZE oder Überschreiben der Belegungspläne geschehen.

FILE NOT FOUND IN DIRECTORY Code D4

Zeigt an, daß kein DIRECTORY-Einsprung im vom DESCRIPTOR angegebenen DIRECTORY- Satz existiert. Die Sätze werden zwar freigegeben, jedoch der DIRECTORY-Einsprung bleibt irgendwo erhalten. Bei Auftreten dieses Fehlers sollte die Diskette (nach Kopie der funktionsfähigen Dateien) neu formatiert werden, um ungewollte Fehler durch evt. Zugriff zu dieser ungültigen Datei zu vermeiden.

FILE ALREADY OPEN (ON ANOTHER UNIT) Code D6

Entsteht beim Versuch des Löschens einer Datei, die bei einer anderen "logischen Funktion" aktiviert ist. Es wird keine Aktion durchgeführt.

8.2.6.12. READ AND DELETE

Vektor: logische Funktion
Code - 1C oder 1D
Datenstartadresse - ab dieser Adresse werden die gelesenen Daten abgespeichert
Datenblocklänge - Anzahl der zu lesenden Bytes; wird auf eine volle Satzlänge aufgerundet, enthält bei Rückkehr die Anzahl der tatsächlich gelesenen Bytes.
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge-Vektor - Enthält bei wahlfreiem Zugriff (RANDOM I/O) in einem Drei-Byte-Feld die Diskettenadresse des ersten gelesenen Satzes.

Aktion: Dieser Aufruf ist nützlich, wenn Daten gelesen verändert und neu gespeichert werden. Vom Satz beginnend, nach dem gerade markierten, werden die Daten aus der Datei in den Speicher transportiert und gleichzeitig gelöscht (DEALLOCATION der Sätze). Der markierte Satz bleibt unverändert, ebenso der Zeiger; dadurch besteht die Möglichkeit, von der gleichen Stelle an neue Daten wieder einzuschreiben.

Der Satz nach dem letzten gelesenen und gelöschten Satz bleibt als 'NEXT RECORD' markiert.

Mögliche Fehler: Alle Disketten-Fehler sind möglich, inclusive PROTECTION.

END OF FILE Code C8
Der letzte Satz wurde gelesen, ohne die gewünschte Byte-Anzahl zu erreichen. Die tatsächliche Zahl wird beim Rücksprung im Datenblocklänge-Feld geliefert, der NEXT-RECORD-Zeiger enthält Ø.

POINTER ERROR Code CA

Der Datentransfer wird beim Auftreten des Zeigerfehlers beendet. Das Datenblocklänge-Feld enthält die Zahl der bis dahin übertragenen Bytes.

FILE NOT OPEN Code CB
Die aufgerufene "logische Funktion" ist nicht aktiviert.

8.2.6.13. READ CURRENT

Vektor: logische Funktion
Code - 1E oder 1F
Datenstartadresse - Daten werden ab dieser Adresse abgespeichert
Datenblocklänge - Anzahl der zu übertragenden Daten. Falls Ø, erfolgt keine Aktion, ansonsten wird ein Satz übertragen. Die tatsächlich übertragene Anzahl ist nach Rücksprung hier enthalten
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge- Vektor - enthält nur bei wahlfreiem Zugriff (RANDOM I/O) in einem Drei- Byte- Feld die Adresse des momentanen Satzes. Ansonsten nicht verwendet.

Aktion: Die Daten des **gerade** markierten Satzes (CURRENT RECORD) werden in den Speicher transferiert. Der Zeiger bleibt unverändert.

Mögliche Fehler: Alle Disk-Fehler außer PROTECTION.

POINTER ERROR Code CA
Entsteht bei falschem Inhalt des PREVIOUS-RECORD-Zeiger. **Der Datentransfer erfolgt trotzdem.**

FILE NOT OPEN Code CB
Die aufgerufene "logische Funktion" ist nicht aktiviert. Es erfolgt kein Transfer.

BEGINNING OF FILE Code D5
Der Zeiger weist auf den nicht lesbaren DESCRIPTOR. Es erfolgt kein Datentransfer und eine Null-Adresse wird bei wahlfreiem Zugriff (RANDOM I/O) zurückgegeben.

8.2.6.14. READ PREVIOUS

Vektor:

logische Funktion	
Code	- 2Ø oder 21
Datenstartadresse	- Beginn des Datenspeicherbereichs
Datenblocklänge	- Anzahl der zu lesenden Bytes; falls =Ø, erfolgt keine Aktion, ansonsten wird ein Satz übertragen. Nach dem Rücksprung ist hier die Zahl der tatsächlichen transferierten Bytes zu finden.
Rückkehradresse	
Fehler-Rückkehradresse	
Fertigstellungscode	
Folge-Vektor	- Nur bei wahlfreiem Zugriff (RANDOM I/O) sollte dieser Vektor einen Drei-Byte-Bereich angeben, der nach Rücksprung die Diskettenadresse des vorhergehenden Satzes ('PREVIOUS RECORD') enthält, ansonsten nicht verwendet.

Aktion: Vorausgesetzt, die Datenblocklänge ist ungleich Ø, wird der vorhergehende Satz des momentan markierten Satzes gelesen. Grundsätzlich wird der Satzzeiger (RECORD POINTER) um einen Satz rückwärts positioniert.

CURRENT RECORD ⟶ NEXT RECORD
PREVIOUS RECORD ⟶ CURRENT RECORD
der dem PREVIOUS RECORD vorhergehende Satz ⟶ PREVIOUS RECORD

Mögliche Fehler: Alle Diskettenfehler außer PROTECTION.

POINTER ERROR Code CA
Entsteht, falls der Vorwärtszeiger des PREVIOUS RECORD nicht auf den aktuellen zeigt.
Die Daten werden trotzdem transferiert.

FILE NOT OPEN Code CB
Die aufgerufene "logische Funktion" ist nicht aktiviert, keine Aktion.

BEGINNING OF FILE Code D5
Es wird versucht, den DESCRIPTOR zu lesen. Dabei wird die Diskettenadresse Ø zurückgegeben.

8.2.6.15. READ DIRECT

Vektor: logische Funktion
Code - 22 oder 23
Datenstartadresse - Anfangsadresse des Datenspeichers
Datenblocklänge - Anzahl der zu transferierenden Bytes, wird auf einen vollen Satz gerundet. Enthält bei Rücksprung die Zahl der tatsächlich transferierten Bytes.
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge-Vektor - Enthält ein Drei-Byte-Feld mit der Diskettenadresse des ersten zu lesenden Satzes. Das dritte Byte ist Ø. Das aufrufende Programm muß sicherstellen, daß die Adresse zur aufgerufenen Datei gehört.

Aktion: Der aufgerufene Satz wird wie der 'nächste Satz' (NEXT RECORD) gelesen. Falls seine Daten nicht die gesamte Datenblocklänge liefern, werden auch die nachfolgenden Sätze gelesen, jedoch alle ohne Zeiger-Kontrolle (POINTER CHECK).
Der Aufruf ist nur bei Dateien mit wahlfreiem Zugriff (OPENING FOR RANDOM I/O) zulässig.

Mögliche Fehler:

INVALID REQUEST Code C1
Entsteht, wenn die Datei **nicht für wahlfreien Zugriff eröffnet ist.**

Alle Fehler des Aufrufs READ BINARY.

8.2.6.16. SKIP FORWARD

Vektor: logische Funktion
Code - 24 oder 25
Datenstartadresse - nicht verwendet
Datenblocklänge - Anzahl der Sätze, die übersprungen werden. Bei Rückkehr ist hier die Anzahl der tatsächlich übersprungenen Sätze enthalten.
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge-Vektor - muß bei wahlfreiem Zugriff (RANDOM I/O) ein Drei-Byte-Feld, in dem die Diskettenadresse des ersten übersprungenen Satzes nach Rückkehr zu finden ist sein.

Aktion: Alle Diskettenfehler, außer PROTECTION.

END OF FILE Code C3
Erreichen des letzten Satzes während der Zeigerbewegung. Der Zeiger markiert diesen; der Zeiger für den nächsten Satz (NEXT RECORD POINTER) ist Null.

POINTER ERROR Code CA
Dieser Fehler kann während des Laufes über die einzelnen Sätze entstehen. In diesem Fall bleibt

der Zeiger 'CURRENT RECORD' auf dem letzten fehlerfreien Satz stehen.

FILE NOT OPEN Code CB
Die aufgerufene 'logische Funktion' ist nicht aktiviert.

8.2.6.17 SKIP BACKWARD

Vektor: logische Funktion
Code - 26 oder 27
Datenstartadresse - nicht verwendet
Datenblocklänge - Anzahl der zu überspringenden Sätze. Bei Rückkehr die tatsächlich übersprungene Zahl.
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge-Vektor - Enthält nur bei wahlfreiem Zugriff (RANDOM I/O) die Diskettenadresse des vorhergehenden Satzes nach der Rückkehr.

Aktion: Der Zeiger auf den aktuellen Satz (CURRENT RECORD POINTER) wird um die gewünschte Zahl rückwärts bewegt kein Datentransfer.

Mögliche Fehler: Alle Diskettenfehler, außer PROTECTION.

POINTER ERROR Code CA
Der Fehler kann während des Laufes erkannt werden. Der Zeiger bleibt auf dem letzten fehlerfreien Satz stehen.

BEGINNING OF FILE Code D5
Der Anfang der Datei wurde vor Ablauf der Satzzahl erreicht. Der Zeiger bleibt am DESCRIPTOR stehen.

8.2.6.18. SKIP TO END

Vektor: logische Funktion
Code - 28 oder 29
Datenstartadresse - nicht verwendet
Datenblocklänge - nicht verwendet
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge-Vektor - zeigt auf ein Drei-Byte-Feld, das nach Rückkehr die Diskettenadresse des letzten Satzes enthält, ansonsten nicht verwendet.

Aktion: Positioniert den 'CURRENT RECORD'-Zeiger auf den letzten Satz.

Mögliche Fehler: Alle Diskettenfehler, außer PROTECTION.

POINTER ERROR Code CA
Zeigerfehler beim letzten Satz (NEXT RECORD POINTER≠Ø); damit Anzeige, daß der letzte lesbare Satz nicht der letzte Satz der Datei war.

FILE NOT OPEN Code CB
Die aufgerufene "logische Funktion" war nicht aktiviert.

8.2.6.19 RENAME

Vektor: logische Funktion
Code - 2A oder 2B
Datenstartadresse - nicht verwendet
Datenblocklänge - nicht verwendet
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge-Vektor - Zeiger zu Bereich mit folgendem Inhalt:

1.Byte - Namenslänge
2.und weitere Bytes - neuer Name

Aktion: Die aufgerufene Datei erhält den neuen Namen aus dem Folge-Vektor. Die Datei muß eröffnet sein (OPEN) oder eine gültige Zuweisung (ASSIGN) muß existieren, damit sie eröffnet werden kann. Falls noch nicht eröffnet, wird sie aktiviert, der DIRECTORY-Einsprung entfernt und ein neuer erzeugt (unter dem neuen Namen). Anschließend wird die Datei (falls nicht bereits eröffnet) wieder geschlossen.
Eine offene SCRATCH-Datei kann zu einer normalen Datei (ein DESCRIPTOR wird erzeugt) , aber eine normale Datei nicht in eine SCRATCH-Datei umgewandelt werden.

Mögliche Fehler: Alle Diskettenfehler sind möglich.

FILE NOT FOUND Code C7
Die aufgerufene Funktion war nicht eröffnet und die zugeordnete Datei ebenfalls; Keine Aktion.

POINTER ERROR Code CA
Der Rückwärtszeiger im DESCRIPTOR zeigt nicht zum DIRECTORY; Keine weitere Aktion.

DUPLICAT FILE Code DØ
Eine Datei gleichen Namens existiert bereits; keine Aktion.

DISK FULL Code D3
Ein neuer DIRECTORY-Satz war nötig,um den Namen aufzunehmen und die Diskette enthielt nicht mehr ausreichend Platz; Keine Aktion.

FILE NOT IN PROPER DIRECTORY RECORD Code D4
Kein DIRECTORY-Einsprung existiert; Keine Aktion.

FILE ALREADY OPEN IN ANOTHER UNIT Code D6
Die aufgerufene Datei ist bereits auf einer anderen 'logischen Funktion' eröffnet; Keine Aktion.

INVALID RENAME Code D7
Versuch der Erzeugung einer SCRATCH-Datei aus einer normalen Datei, bzw. Dateiname mit mehr als 32 Zeichen; Keine Aktion.

8.2.6.20 UPDATE

Vektor: logische Funktion
Code - 2C oder 2D
Datenstartadresse - Adresse der Eigenschaften (Format sh. OPEN-Aufruf)
Datenblocklänge - Zahl der Bytes der Eigenschaften-Information.
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge-Vektor - keiner

Aktion: Bei Änderung der Datei oder ihrer Eigenschaften erfolgt ein Aufruf des DESCRIPTORS, Lesen, Änderung und Rückschreiben, ebenso ein Rückschreiben des Belegungsplans (ALLOCATION MAP). Die Änderung mittels UPDATE erfolgt wie bei CLOSE. Die Datei bleibt aktiv.

Mögliche Fehler: Alle Diskettenfehler.

POINTER ERROR Code CA
Entsteht, falls der DESCRIPTOR-Zeiger nicht auf das DIRECTORY weist.

FILE NOT FOUND Code CB
Die aufgerufene 'logische Funktion' ist nicht aktiviert.

WRONG DISKETTE Code D1
Die Diskettenidentifikation (ID) des Laufwerkes für diese Datei ist nicht mit der im Speicher abgelegten ID identisch. Zeigt zumeist, daß ein Austausch der Disketten oder ein Überschreiben des Belegungsplans stattfand. Keine Aktion.

INVALID ATTRIBUTES Code D2
Hier werden die Eigenschaften TYPE und SATZLÄNGE (RECORD LENGTH) geprüft. Die fehlerhaften Eigenschaften bleiben unverändert.

PROPERTY PROTECTION Code D8
Blockierte Eigenschaften (LOCKED ATTRIBUTES) ,d.h. die Eigenschaften der Datei sind nicht änderbar.

8.2.6.21 SET ATTRIBUTES

Vektor: logische Funktion
Code - 2E oder 2F
Datenstartadresse - Adresse der zu übertragenden Eigenschaften (sh. Format des OPEN-Aufrufs)
Datenblocklänge - Anzahl der Bytes der Eigenschaften.
Rückkehradresse
Fehler-Rückkehradresse
Fertigstellungscode
Folge-Vektor - keiner

Aktion: Der DESCRIPTOR wird gelesen und mit den neuen Eigenschaften versehen, zurückgeschrieben.

Mögliche Fehler: Alle Diskettenfehler.

POINTER ERROR Code CA
Der DESCRIPTOR-Zeiger weist nicht zum DIRECTORY.

FILE NOT FOUND Code CB
Die aufgerufene 'logische Funktion' ist nicht aktiviert.

INVALID ATTRIBUTES Code D2
SATZLÄNGE oder TYP sind ungültig.

PROPERTY PROTECTION CODE Code D8
Blockierte Dateieigenschaften.

8.2.6.22. QUERY ATTRIBUTES

Vektor: logische Funktion
Code - 30 oder 31
Datenstartadresse - Startadresse zum Einschreiben der Eigenschaften in den Speicher.
Datenblocklänge - Hier wird die Anzahl der Bytes der Eigenschaften zurückgegeben.

Rückkehradresse
Fehler-Rückkehradresse
Folge-Vektor - keiner

Aktion: **Lesen des DESCRIPTORs** zur Information über eine eröffnete Datei. Format siehe OPEN-Aufruf.

Mögliche Fehler: Alle Diskettenfehler, außer PROTECTION.

POINTER ERROR Code CA
Der DESCRIPTOR-Zeiger weist nicht zum DIRECTORY.

FILE NOT OPEN Code CB
Die aufgerufene 'logische Funktion' war nicht aktiviert.

ATTRIBUTE LIST TRUNCATED Code 84
WARNUNG! Es wurden mehr als 116 Bytes verlangt, aber nur 116 können transferiert werden.

8.2.7. Programmierung und Nutzung zusätzlicher Gerätetreiber

Zu den zum Betriebssystemkern gehörenden Gerätetreibern (CON, ZDOS) lassen sich noch beliebig andere hinzufügen. Dazu sind zwei Arbeitsschritte notwendig.

8.2.7.1. Programmierung

Gerätetreiber sind Unterprogramme und müssen als Dateien vom Typ Prozedur mit dem Subtyp 1 (P1) auf der Diskette abgelegt sein. Die Parameterübergabe wird über einen Vektor (Datenbereich mit Parametern), dessen Anfangsadresse im IY-Register übergeben wird, sichergestellt.

Die Bedeutung der einzelnen Byte ist in der ZDOS-Beschreibung genau dargelegt. Vor der Programmierung des Steuerprogramms muß als erstes festgestellt werden, ob ein Dateisystem (z.B. MB, MBK, DPÜ) verwaltet werden muß oder ob nur Datentransfer (z.B. LB, LK, Dr) ausreicht.

Steuerprogramme des ersten Typs, die als "Master"-Device (Geräte, auf denen Programme und Daten verzeichnisverwaltet abgelegt werden können) dienen, müssen alle Anforderungen (Requests), die auch das ZDOS bearbeitet, behandeln. Dafür ist es notwendig, zunächst eingehend die ZDOS-Beschreibung zu studieren.

Treiber des zweiten Typs hingegen sind wesentlich einfacher zu erstellen. Hier ist es möglich, auf die meisten Requests zu verzichten und nur einige zu behandeln. Der Konsoltreiber kann dabei als Vorbild dienen.

Das Beispiel des angeführten Druckertreibers ist auf ein Minimum an Software reduziert und ermöglicht somit leicht das Einarbeiten in die Problematik. Das Einfügen der Bedienung anderer Requests, die Möglichkeit der Arbeit im zeitparallelen Betrieb oder das Anpassen an andere Hardware läßt sich dann schrittweise durchführen.

Zunächst werden funktionell nur die Schreibrequests behandelt. Dabei gilt für:

WRBIN:
Ausgabe der angegebenen Zeichenfolge entsprechender Länge. Abgebrochen wird, wenn vorher ein "EOF" (OPPH) in der Folge erkannt wurde. Der Ausgabe des Steuerzeichens "CR" folgt zusätzlich ein "LF".
Die Tabulierung wird softwaremäßig entsprechend dem Tabulatorsatz des Konsoltreibers durchgeführt (bei LB nicht notwendig).

WRLIN:
Gleiche Reaktion wie bei WRBIN, jedoch ist das Steuerzeichen "CR" im Zeichenstrom eine zusätzliche Abbruchbedingung.

WRABS:
Das Datenfeld wird entsprechend der Länge ohne vorherige Abbruchbedingung ausgegeben. Der Tabulator wird nicht interpretiert.
Wichtig ist weiterhin die Festlegung der E/A-Adressen. Für den Anwender gilt, daß er eigene Hardware auf E/A-Adressen ≥80H legen muß, da durch den Bürocomputer die meisten E/A-Adressen im Bereich zwischen 00H...7FH belegt sind.

8.2.7.2. Programmbeispiel anhand des PRINTER-Treibers

```
                                PRINTER                                  PAGE   1
LOC   OBJ CODE M STMT SOURCE STATEMENT                                    ASM 5.9

                    ;*****************************************************
                    ;PRINTER DRIVER
                    ;FUER :DARO 1152 UEBER PIO II
                    ;      DARO 1152 UEBER IFSS
                    ;      DARO 1157 UEBER IFSS
                    ;
                    ;SOFTWARESCHNITTSTELLE IST VECTOR, DESSEN ANFANGSADRESSE
                    ;IM IY-REGISTER UEBERGEBEN WIRD
                    ; (IY+1)          REQUESTCODE
                    ; (IY+2),(IY+3)   HS-ANFANGSADRESSE
                    ; (IY+4),(IY+5)   LAENGE
                    ; (IY+6),(IY+7)   COMPLETION RETURNADRESSE
                    ; (IY+8),(IY+9)   ERROR RETURNADRESSE
                    ; (IY+10)         RETURNCODE
                    ;
                    ;DER TREIBER ARBEITET NICHT ZEITPARALLEL.
                    ;DIE RUECKGABE DER STEUERUNG ERFOLGT ERST NACH
                    ;VOLLSTAENDIGER REQUESTABARBEITUNG.
                    ;*****************************************************
                    ;
0000  FD7E01        PRINTER LD A,(IY+1)
0003  CB87                  RES 0,A
0005  FE00          RQTEST  CP RQINI          ;REQEST AUSTESTEN
0007  2839                  JR Z,OPEN
0009  FE02                  CP RQASN
000B  2835                  JR Z,OPEN
000D  FE04                  CP RQOPEN
000F  2831                  JR Z,OPEN
0011  FE06                  CP RQCLO
0013  281D                  JR Z,OCRET
0015  FE0E                  CP RQWRBI
0017  283D                  JR Z,WR
0019  FE10                  CP RQWRLI
001B  2839                  JR Z,WR
001D  FE48                  CP RQWRAB
001F  2835                  JR Z,WR
0021  FD360AC1              LD (IY+10),CCIO ;FALSCHER REQUEST GEFORDERT
0025  1800                  JR ERRET
                    ;
0027  FD6609        ERRET   LD H,(IY+9)       ;RUECKKEHR BEI FEHLER
002A  FD6E08                LD L,(IY+8)
002D  7C                    LD A,H
002E  B5                    OR L
002F  2805                  JR Z,COMRET       ;ERROR-ADRESSE WAR 0
0031  E9                    JP (HL)
0032  FD360A80      OCRET   LD (IY+10),OCOC   ;BEARBEITUNG FERTIG
0036  FDCB0146      COMRET  BIT 0,(IY+1)      ;RETURN BEI UNGER. REQUEST
003A  C8                    RET Z
003B  FD6607                LD H,(IY+7)
003E  FD6E06                LD L,(IY+6)
0041  E9                    JP (HL)           ;ZUR COMPLET-ADRESSE
                    ;
                    ;REQUESTBEARBEITUNG
                    ;
0042  010000        OPEN    LD BC,0
0045  DB56          RDY     IN A,(PBD)        ;TEST OB DRUCKER BETRIEBSBEREIT
0047  CB4F                  BIT 1,A
0049  20E7                  JR NZ,OCRET
```

```
LOC   OBJ CODE  M STMT SOURCE STATEMENT

004B  0D          59          DEC C
004C  20F7        60          JR NZ,RDY
004E  10F5        61          DJNZ RDY
0050  FD360AC2    62          LD (IY+0AH),CCHR        ;NICHT BEREIT
0054  18D1        63          JR ERRET
                  64  ;
                  65  ;DATENAUSGABE ABSOLUT, BINAER, ZEILENWEISE
                  66  ;
0056  FD4605      67  WR      LD B,(IY+5)             ;BC:=LAENGE
0059  FD4E04      68          LD C,(IY+4)
005C  79          69          LD A,C
005D  B0          70          OR B
005E  28D2        71          JR Z,OCRET              ;LAENGE WAR 0
0060  1600        72          LD D,0                  ;RESET ZEICHENZAEHLER
0062  1E00        73          LD E,0
0064  211201      R 74        LD HL,POS               ;RESET SPALTENZAEHLER
0067  3600        75          LD (HL),0
0069  FD6603      76          LD H,(IY+3)             ;HL:=ADRESSE
006C  FD6E02      77          LD L,(IY+2)
006F  FD7E01      78  WRCYC   LD A,(IY+1)             ;AUSGABEZYKLUS
0072  CB87        79          RES 0,A
0074  FE46        80          CP RQWRAB
0076  7E          81          LD A,(HL)
0077  2811        82          JR Z,WRCHR              ;KEINE EOD U. TAB AUSWERTUNG BEI ABS
0079  FEF2        83          CP EOF
007B  2009        84          JR NZ,TAB?
007D  13          85  EOD     INC DE
007E  FD7205      86          LD (IY+5),D             ;LAENGE RUECKSPEICHERN
0081  FD7304      87          LD (IY+4),E
0084  18A0        88          JR OCRET
0086  FE09        89  TAB?    CP TAB
0088  28EA        90          JR Z,TABBE
008A  CDF100      R 91  WRCHR   CALL SDA
008D  E5          92          PUSH HL
008E  211201      R 93          LD HL,POS
0091  34          94          INC (HL)                ;SPALTENNUMMER ERHOEHEN
0092  E1          95          POP HL
0093  FE0D        96          CP CR
0095  2013        97          JR NZ,UPDAT
0097  CDCB00      R 98          CALL SDAP             ;AUSGABE LF UND UNTERBRECHUNGSPENDING
009A  E5          99          PUSH HL
009B  211201      R 100         LD HL,POS
009E  3600        101           LD (HL),0             ;SPALTENNUMMER:=0 NACH CR
00A0  E1          102           POP HL
00A1  FD7E01      103           LD A,(IY+1)
00A4  CB87        104           RES 0,A
00A6  FE10        105           CP BQWRLI
00A8  28D3        106           JR Z,EOD              ;ENDE BEI ZEILENAUSGABE
00AA  23          107 UPDAT     INC HL                ;ERHOEHEN ADRESSE
00AB  0B          108           DEC BC                ;VERMINDERN LAENGE
00AC  13          109           INC DE                ;ERHOEHEN ZEICHENZAEHLER
00AD  79          110           LD A,C
00AE  B0          111           OR B
00AF  20BE        112           JR NZ,WRCYC           ;LAENGE=0 ?
00B1  C33200      R 113         JP OCRET
                  114 ;
00B4  C5          115 TABBE     PUSH BC               ;TABULATORBEHANDLUNG
00B5  E5          116           PUSH HL
```

```
00B6  3E20          117  TABBEM:  LD A,SPACE
00B8  CDF100    R   118           CALL SDA
00BB  211201    R   119           LD HL,POS
00BE  34            120           INC (HL)
00BF  48            121           LD C,(HL)
00C0  0600          122           LD B,0
00C2  21C424        123           LD HL,TABTAB
00C5  09            124           ADD HL,BC
00C6  7E            125           LD A,(HL)
00C7  A7            126           AND A                ;IST TABULATORPOSITION GESETZT ?
00C8  28EC          127           JR Z,TABBEM
00CA  E1            128           POP HL
00CB  C1            129           POP BC
00CC  13DC          130           JR UFDAT
                    131  ;
00CE  3E0A          132  SDAP     LD A,LF              ;AUSGABE "LF" NACH "CR"
00D0  CDF100    R   133           CALL SDA
00D3  C5            134           PUSH BC
00D4  D5            135           PUSH DE
00D5  E5            136           PUSH HL
00D6  FDE5          137           PUSH IY
00D8  CD0610        138           CALL TTYST           ;"ESC"-PENDING
00DB  FDE1          139           POP IY
00DD  E1            140           POP HL
00DE  D1            141           POP DE
00DF  C1            142           POP BC
00E0  3E0D          143           LD A,CR
00E2  C0            144           RET NZ
00E3  E1            145           POP HL               ;RUECKKEHRADRESSE VERNICHTEN
00E4  FD7304        146           LD (IY+4),E
00E7  FD7205        147           LD (IY+5),D
00EA  FD360A49      148           LD (IY+10),0CAH ;RET FUER "ESC"
00EE  C32700    R   149           JP ERRET
                    150  ;
                    151  ;EINZELZEICHENDRIVER FUER 1152
                    152  ;AUSGABE EINES ZEICHENS UEBER IFSS-SCHNITTSTELLE (SIO).
                    153  ;DABEI SIGNALISIERT "DC3" DAS STOPPEN UND "DC1" DAS
                    154  ;FORTSETZEN DER AUSGABE.
                    155  ;
                    156  ;BEI PIO II-SCHNITTSTELLE WIRD DIE SYNCHRONISATION DURCH
                    157  ;ABFRAGE DER SENDEBEREITSCHAFT DES DRUCKERS 1152 ERREICHT.
                    158  ;
                    159  ;PIO, SIO UND CTC WURDEN SCHON BEIM SYSTEMANLAUF
                    160  ;INITIALISIERT.
                    161  ;
00F1  F5            162  SDA      PUSH AF
00F2  DB5E          163  IFSS     IN A,(SBD)
00F4  F613          164           CP DC3               ;DRUCKERPUFFER VOLL ?
00F6  2006          165           JR NZ,SSA
00F8  DB5E          166           IN A,(SBD)
00FA  F691          167           CP DC1               ;DRUCKERPUFFER FREI ?
00FC  20FA          168           JR NZ,a-4
00FE  DB57          169  SSA      IN A,(SBC)
0100  CB57          170           BIT 2,A              ;SENDEPUFFER LEER ?
0102  28FA          171           JR Z,a-4
0104  DB56          172  SDAX     IN A,(PBD)
0106  CB4F          173           BIT 1,A              ;DRUCKER SENDEBEREIT ?
0108  28FA          174           JR Z,SDAX
```

```
010A DB54          175             IN A,(PAD)
010C F1            176             POP AF
010D D35B          177             OUT (SSD),A     ;ZEICHEN --> SIO
010F D354          178             OUT (PAD),A     ;ZEICHEN --> PIO
0111 C9            179             RET
                   180 ;
                   181 ;DATEN
0112               182 POS     DEFS 1              ;TABULATORPOSITIONSZAEHLER
                   183 ;
                   184 ;ASCII-SONDERZEICHEN
                   185 TAB     EQU 9               ;TABULATOR
                   186 LF      EQU 0AH             ;ZEILENSCHALTUNG
                   187 CR      EQU 0DH             ;WAGENRUECKLAUF
                   188 DC1     EQU 91H
                   189 DC3     EQU 13H
                   190 SPACE   EQU 20H             ;LEERZEICHEN
                   191 EOF     EQU 0FFH            ;DATEI ENDE
                   192         ;
                   193         ;I/O-REQUESTS
                   194 RQINI   EQU 0               ;INITIALIZE
                   195 RQASN   EQU 2               ;ASIGN
                   196 RQOPEN  EQU 4               ;OPEN
                   197 RQCLO   EQU 6               ;CLOSE
                   198 RQWRBI  EQU 0EH             ;WRITE BINAER
                   199 RQWRLI  EQU 0CH             ;  "   LINE
                   200 RQWRAB  EQU 48H             ;  "   ABSOLUT
                   201 ;
                   202 ;COMPLETIONCODES
                   203 CCOO    EQU 80H             ; OK
                   204 CCIO    EQU 0C1H            ;INVALID REQEST
                   205 CCNR    EQU 0C2H            ;DEVICE NOT READY
                   206 CCAB    EQU 049H            ;TRANSFER ABORT
                   207 ;
                   208 ;SYSTEMRESOURCEN
                   209 TTYST   EQU 1006H           ;UP FUER TASTATURSTATUS
                   210 TABTAB  EQU 24C4H           ;TABULATORFELD DES KONSOLTREIBERS
                   211 ;
                   212 ;PORTADRESSEN
                   213 PAD     EQU 54H             ;DATENPORT PIO
                   214 PBD     EQU 56H             ;STEUERPORT PIO
                   215 SBD     EQU 58H             ;DATENPORT SIO
                   216 SBC     EQU 5AH             ;STEUERPORT SIO
```

8.2.7.3. Nutzung allgemein

Nachdem durch einen Übersetzungs- und Bindevorgang (Zweckmäßigerweise sind zusätzliche Treiber in den hinteren Teil des Arbeitsspeichers zu binden, unmittelbar vor den Bildwiederholspeicher und die Diskettenbelegungsmaps!) ein lauffähiges Maschinenprogramm erzeugt wurde, ist sein Subtyp durch das UDOS-Kommando

```
SET SUBTYPE OF "FILENAME" TO 1
```

auf 1 zu setzen. Dieser Parameter kennzeichnet Treiberprogramme. Durch das Kommando "ACTIVATE" kann der Treiber in das System eingebunden und durch "DEACTIVATE" wieder herausgelöst werden. Der Datentransfer ist dann durch die entsprechenden Dienstprogramme, wie "MOVE", "COPY" möglich. Mit Hilfe des Kommandos "DEFINE" ist es auch möglich, einem Treiber eine oder mehrere logische Gerätenummern zuzuweisen. Damit kann dieses dann auch programmäßig angesprochen werden. Da diese zusätzlichen Steuerprogramme meist nur zeitweilig genutzt werden, ist der Nutzungsvorgang günstig in einer Kommandodatei zusammenfaßbar.

Beispiele:

1. Drucken einer Textdatei

   -UDOS-Aufruf:

   ```
   %DO PWRITE TEXT
   ```

   -Kommandodatei PWRITE:

   ```
   ACTIVATE $PRINTER
   COPY #1 $PRINTER
   DEACTIVATE $PRINTER
   ```

2. Drucken des Inhaltes einer Datei hexadezimal:

   -UDOS-Aufruf:

   ```
   %DO PDUMP DATEI
   ```

   -Kommandodatei PDUMP:

   ```
   ACTIVATE $PRINTER
   DEFINE SYSLST $PRINTER
   DUMP #1
   DEFINE 3 $CON
   DEACTIVATE $PRINTER
   ```

   (#n wird durch den jeweiligen n-ten Parameter des DO-Kommandos nach dem Kommandodateinamen ersetzt.)

8.2.7.3.1. Druckertreiber SD

Diese Treiberroutine dient der formatierten Ausgabe von Texten. Der Treiber ist fuer die Drucker SD 1152 und SD 1157 ueber PIO II - Schnittstelle der Steckeinheit K 6028 oder IFSS - Schnittstelle der Steckeinheit K 8025 nutzbar. Durch Hinzufuegen von Optionen beim Aktivieren des Treibers ist es moeglich, eine Seitenformatierung bei der Ausgabe von Texten zu erzeugen. Werden keine Optionen angegeben, arbeitet der Treiber wie bisher (PRINTER) ohne eigene Seitenformatierung (Nutzung bei Assemblerlisten).

Folgende Optionen sind moeglich:

SL= xx	Anzahl der Zeilen/Seite #> keine formatierte Ausgabe
N=Y	Mit Seitennumerierung (1.99) #> keine Seitennumerierung
FSN=xx	Nummer der ersten Seite #> Nummer der ersten Seite= 1
LFC=x	Anzahl der LF/Zeile (mehrzeilig) #> Anzahl der LF/CR=1 (einzeilig)
FF=Ax	Format x=3,4,5 ==> A3, A4, A5 #> Format A4
TL=Y	Mit Trennlinie zwischen den Seiten Formularvorschub softwaremaeszig #> ohne Trennlinie, Formularvorschub wird durch Drucker gesteuert
HL='Abcd..'	Kopfzeile bis 32 Zeichen #> keine Kopfzeile
#> :	Option nicht vorhanden

Beispiel fuer die Nutzung:

```
%ACTIVATE ¤SD N=Y SL=66 TL=Y
%COPY TEXT ¤SD
```

ACHTUNG: Wenn der Treiber SD benutzt wird, darf die Monitortaste (bzw. Taste OFF bei der Tastatur K7634) nicht betaetigt werden !!!

==> Ein Text wird im Format A4 mit 66 Zeilen/Seite numeriert ausgegeben. Zwischen den Seiten befinden sich Trennlinien. (Fuer Rollenpapier)
Die Arbeit mit Kommandodateien ist weiterhin moeglich.

Der Druckertreiber SD kann auf andere Adressen gebunden werden, indem folgende Kommandos ausgefuehrt werden:

```
%LINK ¤=Adresse SD SDH (E=SD NOM ST=0)
%SET SUBTYPE OF SD TO 1
```

8.2.7.3.2. Lochbandtreiber PTAPE

Zur Bedienung der Lochbandeinheit K 6200 ueber die StE K 6025 werden der Lochbandtreibermodul PTAPE, die Lochbandaufbereitungsprogramme TAPE.READ und TAPE.WRITE sowie die zwei Kommandodateien TREAD und TWRITE zur Verfuegung gestellt.
Der Lochbandtreiber ist konsoltypisch und realisiert die folgenden Requests:

INIT, ASSIGN, OPEN, CLOSE, DEACTIVATE,
READ BINARY, WRITE BINARY,
READ LINE, WRITE LINE.

Es kann damit fuer die Uebertragung von Textdateien (Typ A) in Verbindung mit den UDOS-Dienstprogrammen wie PRINT, COPY genutzt werden. Vor der Benutzung ist der Treiber zu aktivieren. Guenstiger ist es aber, den Treiber in Verbindung mit den Lochbandaufbereitungsprogrammen TAPE.READ und TAPE.WRITE zu benutzen. Diese Module organisieren das automatische Stanzen des Vor- und Nachspanns und die Umwandlung des Floppy-Datenformats in das Lochbandformat. Programme (Typ P) lassen sich dadurch im allgemeinueblichen Intel-Lochbandformat darstellen.

Die Lochbandaufbereitungsprogramme TAPE.READ und TAPE.WRITE benutzen das logische Geraet 20 und benoetigen folgende Parameter:

```
%TAPE.READ Dateiname (F="Format".OR.RL="Recordlaenge".OR.
           E="Entry-Point".OR.0)

%TAPE.WRITE Dateiname (F="Format".OR.RL="Recordlaenge")
```

Die Parameter haben folgende Bedeutung:

1. Dateiname: entsprechend den UDOS-Konventionen

2. Format: Es sind zwei Formate erlaubt.
   F=A (ASCII) Die Daten werden als Zeichenstrom gestanzt bzw. gelesen bis zum Ende der Datei bzw. dem Blockende. Zur Verarbeitung duerfen nur Textdateien (Typ A) gelangen.
   F=I Das Format kann nur fuer Programme (Dateityp P) benutzt werden.

   ACHTUNG: Die Format-Option muss bei der Benutzung des Lochbandtreibers unbedingt angegeben werden !!!

3. Recordlaenge: Beim Einlesen wird die auf Diskette abzuspeichernde Datei in der angegebenen Recordlaenge aufgezeichnet. Dabei sind die durch ZDOS erlaubten Recordlaengen von 80H, 100H, 200H, 400H und 800H moeglich.
   Beim Ausstanzen von Programmen (F=I) kann mit dieser Option die Blocklaenge gewaehlt werden.

4. Entry-Point: Beim Einlesen von Programmen (F=I) kann der Eintrittspunkt fuer den Datei-Descriptor definiert werden. Das ist wichtig, falls der Eintrittspunkt nicht mit dem ersten Befehl des Programmes identisch ist.

5. Overwrite-Option 0: Beim Einlesen wird eine bereits unter dem spezifizierten Dateinamen vorhandene Datei ueberschrieben.

Zur Vereinfachung der Bedienung koennen die beiden Kommandodateien TREAD und TWRITE benutzt werden. Sie haben folgenden Inhalt:

TWRITE:

```
ACTIVATE ¤PTAPE
DEFINE 20 ¤PTAPE
TAPE.WRITE #1 (#2 [#3 ])
DEFINE 20 ¤NULL
DEACTIVATE ¤PTAPE
```

TREAD:

```
ACTIVATE ¤PTAPE
DEFINE 20 ¤PTAPE
TAPE.READ #1 (#2 [#3 [#4 [#5]]])
DEFINE 20 ¤NULL
DEACTIVATE ¤PTAPE
```

Als Fehlermeldungen koennen durch den Treiber geliefert werden:

C1 - Invalid Request
C2 - Device not ready
C4 - Diskettenfehler (Sektor nicht auffindbar)
C5 - " (Spur defekt)
C6 - " (CRC - Fehler)
CA - Pointerfehler
D0 - Datei schon vorhanden
49 - Abbruch

Als EOF-Marke (bzw. Bandende) werden zehn aufeinanderfolgende Zeichen mit dem hexadezimalen Wert 00H interpretiert.

Beispiele fuer die Anwendung:

%DO TWRITE NOTE.TO.PTAPE (F=A)
Die Textdatei NOTE.TO.PTAPE (Typ A) wird auf Lochband ausgestanzt.

%DO TREAD PHASE (F=I RL=100 E=4011)
Das Lochband wird bis zur EOF-Marke (10 Zeichen 00H) eingelesen und unter dem Namen PHASE als Procedure-Datei mit der Recordlaenge 100H und dem Entry-Point 4011H auf Diskette aufgezeichnet.

Der Lochbandtreiber PTAPE kann auf andere Adressen gebunden werden, indem folgende Kommandos ausgefuert werden:

```
%LINK ¤=Adresse PTAPE.OBJ (E=PTAPE NOM)
%SET SUBTYPE OF PTAPE TO 1
```

8.3. EDITOR

Die Erstellung und Modifizierung von Anwenderdateien erfolgt mittels des Dienstprogrammes TEXT EDITOR, das folgende Merkmale besitzt:

- zeilenorientiertes Editieren;
  (Zeilenzähler enthält immer die Nummer der zuletzt zugegriffenen Zeile, Anfangswert = Ø)
- Zeichenkettenbehandlung innerhalb einer Zeile;
- Erzeugung und Bearbeitung von Dateien des Typs A (ASCII);
- Datentransport von und zur Diskette erfolgt automatisch;
- Satzlänge: ≤ 512 Zeichen
- Automatische Ausgabe bzw. Suche des Quellprogramms auf Diskette, wobei das Diskettenlaufwerk Priorität hat, das dem den EDITOR enthaltenden Laufwerk folgt.

Programmaufruf:

```
% EDIT dateiname .S [parameter] ET1
```

- dateiname ≤ 32 Zeichen
- .S - Namenzusatz bedeutet: Quellendatei
- parameter:
  - O = dateiname - Name der Rückkehrkopie
  - N - ohne Rückkehrkopie
  - RL = satzlänge - Satzlänge für neu zu erstellende Datei

Rücksprung ins OS:

```
> QUIT ET1
```

Meldungen des Editors:

EDIT	- a) angesprochene Datei wurde in Arbeitsspeicher geladen und steht zum Editieren bereit b) nach Abschluß eines Input-Kommandos
NAME ?	- fehlende Namensangabe beim Aufruf des Editors
TYPE	- Angabe einer Dateiart ist erforderlich

NEW FILE	- angesprochene Datei ist noch nicht vorhanden; automatischer Übergang in INPUT-Modus zum Neuerfassen der Datei
INPUT	- Eingabe-Modus
NO CHANGE	- spezifizierter Text zur Änderung nicht gefunden
ERROR Code	- nicht ordnungsgemäße Durchführung einer Disketten- Ein/Ausgabeaktivität; " Code " spezifiziert Fehlerart
STRING NOT IN BLOCK PROCEED	- spezifizierte Zeichenkette nicht gefunden im aktuellen Block; Abfrage, ob im nachfolgenden Block weitergesucht werden soll (Antwort Y oder N)

8.3.1. Erfassen von Dateien

Mittels Programmaufruf: EDIT dateiname wird auf der Diskette gesucht, ob eine Datei mit dem angegebenen Namen vorhanden ist. Wird der Name nicht gefunden, steht der Editor im Eingabe- Modus und es kann mit der Programmerfassung begonnen werden.

```
% EDIT      dateiname .S ET1
  EDIT
  NEW FILE
  INPUT                                  ;Input- Modus
  ..........  ET1
  ..........  ET1        Programmzeilen
  ..........  ET1
        .
        .
  ..........  ET1 ET1    letzte Zeile
                                         ;Sprung in den Editor

  EDIT
  > QUIT                                 ;Sprung ins OS
%
```

8.3.2. Ändern von Dateien

Ist eine Datei mit dem eingegebenen Namen bereits vorhanden, wird sie als Rückkehrdatei kopiert und in den Speicher zum Editieren geladen. Der Rückkehrdatei wird der Dateiname der Originaldatei mit dem Zusatz . OLD zugeordnet.

TEST .S ⟶ TEST.S. OLD

Das Ändern der Datei erfolgt zeilenweise mittels Editier-Kommandos, die im Punkt 8.3.3. beschrieben sind.
Das Einfügen neuer Textzeilen muß im Input- Modus geschehen.

```
% EDIT dateiname .S ET1
  EDIT
  > kommando [zeichenkette]        ; Bearbeitung der Zeile
  > I                              ; Übergang zum Eingabe- Modus
  INPUT                            ; Eingabe- Modus
  ......... ET1 ET1                ; Sprung in Editor- Modus
  EDIT
  > QUIT                           ; Sprung ins OS
%
```

8.3.3. Kommandos des Editors

Die Kommandos des Editors können zusätzlich modifiziert werden:

- durch eine Zahl n; Wiederholung des Kommandos n-mal
  z.B.: Up 10; Zeilenzähler wird um 10 vermindert
  (Voreinstellung von n ist 1)

- durch eine Zeichenkette zwischen Begrenzungszeichen (alle nichtnumerischen Zeichen außer Leerzeichen zulässig); Wiederholung des Kommandos, bis Zeichenkette gefunden wird
  z.B.: PRINT / LD A, (DE); drucke ab aktueller Zeile bis einschließlich Zeile mit: LD A, (DE)

Die Modifikationszeichen sind durch mindestens ein Leerzeichen vom Kommandowort zu trennen.
In Klammern [ ] gesetzte Kommandoteile können wahlweise verwendet werden.
Für den Aufruf eines Kommandos müssen mindestens die unterstrichenen Buchstaben angegeben werden.

Kommandodefinitionen

Again n	Wiederholung des vorhergehenden Kommandos n-mal
Bottom	Zeilenzähler wird auf letzte Zeile der Datei gesetzt und ausgegeben; (im Brief- Modus: Ausgabenunterdrückung)
BRief	Brief-Modus, d.h. die ursprünglich vorgesehene Textausgabe nach: Bottom, Change, Get, Locate, Next und Up erfolgt nicht.

Change / alte zeichenkette / neue zeichenkette /[n1 n2]

Ersetzt die " alte Zeichenkette " durch die " neue Zeichenkette ".

n1= Anzahl Zeilen, in denen die "alte Zeichenkette " gesucht und geändert wird, (bei * statt n1 → Änderung in allen folgenden Zeilen)

n2= Anzahl der in einer Zeile vorkommenden " alten Zeichenketten ", die geändert werden sollen, (bei * statt n2 → Änderung aller in einer Zeile vorkommenden " alten Zeichenketten ")

z.B.: >C /A/D/↵↵ ; jedes ab laufender Zeile vorkommende 'A' wird durch 'D' ersetzt

Sollte die Änderung auch noch in allen vorhergehenden Zeilen durchgeführt werden, muß vor dem 'CHANGE' ein 'TOP'-Kommando stehen.
Die Angabe von n1 und n2 ist optional (Standardwert =1)
Löschen der angegebenen Zeichenfolge mit:
C /zeichenkette//.
'NO CHANGE' wird ausgegeben, wenn der angeführte Text nicht gefunden wurde.
Die Kommando -Abarbeitung kann mit '?' unterbrochen werden.

DElete [n/zeichenkette/]

Löschen von nZeilen der Datei, beginnend mit der aktuellen Zeile.
Bei Angabe einer Zeichenkette werden alle Zeilen bis, aber nicht inklusive, Zeile mit der angeführten Zeichenkette gelöscht.

z.B.: >DE 3 ; 3 Zeilen ab laufender werden gelöscht

>DE /LD A,B ; alle Zeilen ab laufender bis vor die Zeile mit Zeichenkette 'LD A,B' werden gelöscht.

Find / zeichenkette /

Einstellen des Zeilenzählers auf die lt. „Zeichenkette" angeführte Zeile

(z.B. zum Auffinden von Namen in Assembler-programmen).

GEt [dateiname]

Lädt einen auf der Diskette durch ein PUT - Kommando zwischengespeicherten Text -Block in den Arbeitsspeicher; beginnend mit der nächstfolgenden Zeile. Erfolgt keine Namensangabe, wird die durch ein PUT -oder PUTD -Kommando erzeugte temporäre Datei eingefügt.

Goto n Stellt den Zeilenzähler auf Zeilennummer n ein.

Input [textzeile]

Fügt die angegebene Textzeile im Anschluß an die laufende Zeile ein. Ohne sofortige Texteingabe geht der Editor in den Input -Modus, wo mehrere Textzeilen nach der laufenden Zeile eingefügt werden können.
Rücksprung in Editor- Modus mit „ET1 ".

Join &kommando& [kommando &] *

Aufgeführte Kommandos (max. 512 Zeichen) werden mit Tastendruck ET1 sofort ausgeführt.
Zwischen den Kommandos und Begrenzungszeichen sind Leerzeichen nicht erlaubt.
Als Begrenzungszeichen kann jedes beliebige Zeichen verwendet werden, es darf jedoch nicht in den Kommandos selbst auftreten.

z.B.: > J&T&P 6&G 9&D E 2

bedeutet: Zeilenzähler auf Ø setzen,
die ersten 6 Zeilen drucken,
den Zeilenzähler auf Zeile 9 einstellen und zwei Zeilen löschen

LIneno — Gibt die Zeilennummer der laufenden Zeile aus.

Locate — /zeichenkette/

Einstellen des Zeilenzählers auf die Zeile, die der Zeile mit der angegebenen Zeichenkette folgt.

Macro — & kommando& [kommando &]*

Aufgeführte Kommandos werden mit Tastendruck ET1, im Makropuffer abgespeichert. Mit dem Kommando „X" ist die abgespeicherte Kommandofolge jederzeit abarbeitbar.

Next — [n/zeichenkette/]

Zeilenzähler wird um nZeilen erhöht bzw. auf die Zeile mit der entsprechenden Zeichenkette eingestellt.

Print — [n/zeichenkette/]

Gibt ab laufender Zeile die nächsten nZeilen aus oder bis zum ersten Auftreten der angegebenen Zeichenkette. Die Ausgabe kann mit „?" unterbrochen werden.

PUt — [n/zeichenkette/ [dateiname [RL = n]]]

Ausgabe auf eine Diskettendatei mit entsprechendem Dateinamen, spezifiziert durch:

- nZeilen oder
- angegebene Zeichenkette, bis zu dieser (aber nicht einschließlich dieser) Zeile ausgegeben werden soll
- RL (Satzlänge) kann mit max. 200 vereinbart werden

Ohne Namensangabe wird die temporäre PUT/GET - Datei verwendet. Bereits vorhandener Text wird dabei überschrieben.

z.B.: >PU 12 ZWIDAT RL= 100 ; Ausgabe 12 Zeilen in ZWIDAT, SL= 256 Byte

PUTD [n /zeichenkette/ [dateiname]]

Ausgabe auf eine Diskettendatei wie PUT, jedoch werden die überspielten Daten in der Anwenderdatei gelöscht.

QUIT Schließen der Anwenderdatei, Steuerung wird an das OS zurückgegeben.

Replace [textzeile]

Angegebener Text überschreibt den Text in der laufenden Zeile, wobei der Abstand vom Kommando genau ein Leerzeichen betragen muß.

Erfolgt keine Textangabe, geht der Editor in den Input-Modus. Der über Tastatur eingegebene Text wird eingefügt, wobei die laufende Zeile mit dem alten Text gelöscht wird. Mit der Tastaturbedienung „ET1" wird der Input-Modus verlassen.

Top Setzt den Zeilenzähler auf die Zeile Ø, d.h. vor die erste Zeile der Anwenderdatei.

Up [n/zeichenkette]

Verringert den Zeilenzähler um n Zeilen bzw. stellt ihn auf die Zeile mit der spezifizierten Zeichenkette ein. Wird die Zeichenkette nicht gefunden, erscheint die Mitteilung "STRING NOT IN BLOCK"

und das Kommando wird beendet.

Verify Verläßt den Brief-Modus und gibt die Textzeile aus nach den Kommandos:

Bottom, Change, Find, Get, Locate, Next, Up.

Window Zeigt die Zeilennummern der ersten und letzten Textzeile im Editierungs-Puffer an.

Xecute Die mit dem Kommando „M" im Makropuffer gespeicherte Kommandofolge wird abgearbeitet.

8.4. Assembler

Das über den Editor erfaßte Quellprogramm wird mit dem Assembler in einen verschieblichen Objektmodul gewandelt (Absolute Adressierung mit " Absolute " möglich).

Neben der Objektdatei wird gleichzeitig eine Listendatei erstellt, die den Quellcode und den zugehörigen Objektcode ausweisen.
Optional kann eine Symbolübersicht in die Listendatei aufgenommen werden.
Ein vom Rechner abarbeitbares Programm entsteht durch das Zusammenfügen der einzelnen Programmteile aus der Objektdatei mittels Programmverbinder " LINK " .
Über den Linker erfolgt auch eine entsprechende Adressenzuordnung für globale und externe Symbole in verschiedenen Moduln , die einen gegenseitigen Zugriff ermöglichen.

8.4.1. Assemblerbedienung

Programmaufruf

% ASM dateiname. S *[optionen]

- *Angabe mehrerer Dateien möglich; getrennt durch Leerzeichen
- Dateiname muß mit ".S" enden
- Optionen sind wahlfrei und müssen in Klammern () gesetzt werden

Optionen

(Absolute)	; absolute Adressierung des Module, ansonsten verschieblich
(D-zeichenkette)	; angeführte Zeichenkette (max. 18Zeichen) erscheint als Kopfzeile jeder neuen Seite
(Macro)	; ermöglicht Makroverarbeitung, sonst Fehlerausschrift bei Makrodefinitionen und -aufrufen
(NOliet)	; unterbindet Listendatei auf der Diskette

(NOMaclist) ; unterbindet Aufbau und Ausgabe der Makrodatei

(NOObject) ; unterbindet Aufbau der Objektdatei auf der Diskette

(Print) ; bewirkt Listenausgabe auf das Listgerät (Standardfall: Bildschirm)

(Symbol) ; erstellt Tabelle interner Symbole in binärer Darstellung, die an den Objektmodul angefügt wird (sortiertes Listing durch Pass 3)

(Xref) ; erstellt Symbolliste am Ende der Listendatei (sortierte Crossreference -Tabelle durch Pass 3)

Eine Änderung der Namens -bzw. logischen Gerätezuweisung kann mit folgenden Optionen erfolgen:

(I= dateiname) ; Zwischendatei
(L= dateiname) ; Listendatei
(M= dateiname) ; Makrodatei
(O= dateiname) ; Objektdatei
(T= dateiname) ; Symbolüberlaufdatei
(X= dateiname) ; Crossreference -Datei

z.B.: % ASM TEST.S

- erstellt die Dateien TEST. OBJ und TEST.L auf der Diskette mit TEST.S; ohne Listenausgabe

z.B.: % ASM TEST.S (O=Ø / TES1 S P NOL)

- erstellt die Objektdatei "TES1" mit anschließender Symboltabelle auf Diskette im Laufwerk Ø; es erfolgt die Ausgabe auf das Listgerät; der Aufbau der Listendatei auf Diskette wird unterdrückt

Die Quelldateien können sowohl auf der System- als auch auf der Anwenderdiskette stehen. Bei identischen Dateinamen auf beiden Disketten erlangt die Datei auf der Anwenderdiskette den Vorrang.
Der Assemblerlauf erfolgt normalerweise in zwei Etappen (Pass1 und 2)

Pass1:
- Identifikation der Anweisungen
- Prüfung der Anweisung auf formale Richtigkeit
- Ausgabe einer Zwischendatei auf Systemdiskette
- Abschluß: "Pass1 COMPLETE"

Pass2:
- Erstellung der Objektdatei
- Erstellung der Listingdatei

In bestimmten Fällen wird noch ein dritter Pass angeschlossen, in dem folgende Aktionen ablaufen

Pass3:
- Erstellung einer sortierten Symboltabelle (wahlweise)

Am Ende des Assemblerlaufes erscheint die Ausschrift " ASSEMBLY COMPLETE "

8.4.2. Sprachbeschreibung

Die Assemblersprache für das UDOS- System entspricht im allgemeinen der maschinenorientierten symbolischen Programmiersprache CABS 1520, die alle Befehle des U880 beinhaltet. Abweichungen in der mnemonischen Schreibweise der Befehle des UDOS- Assemblers werden im Gliederungspunkt 8.4.2.5. gesondert angeführt.
Die zur Steuerung des Übersetzungsvorganges erforderlichen Pseudobefehle und Kommandos weisen gegenüber CABS ebenfalls Unterschiede auf und werden nachfolgend komplett aufgeführt. Der UDOS -Assembler ermöglicht außerdem die Verarbeitung von Makros(siehe 8.4.2.3.).

8.4.2.1. Ausdrücke

Der Assembler übersetzt sowohl logische als auch arithmetische Ausdrücke und zwar grundsätzlich von links nach rechts.
Für die Operationen ist folgende Reihenfolge festgelegt:

Operator	Funktion	Beispiel
+ -	Vorzeichen pos./neg.	LD B,-16; (=FØH)
**	Potenzieren	

*	Multiplizieren	
/	Dividieren	
.MOD.	Modulo (Div. m. ganzzahligem Ergebnis)	1Ø.MOD.4 ; Ergeb:2
.SHR.	Verschiebung rechts	
.SHL.	Verschiebung links	
+	Addieren	
-	Subtrahieren	
.NOT.oder \	Negation	COND.NOT. ('ERG'=' ')
.AND.oder &	Log. UND	
.OR.oder ^	Log. ODER	
.XOR.	Exclusiv ODER	
.EQ.oder=	Gleich	2.EQ.1
.GT.oder >	Größer als	
.LT.oder <	Kleiner als	COND ERG <Ø
.UGT.	Vorzeichenloses " Größer als "	
.ULT.	Vorzeichenloses " Kleiner als "	
.RES.	Resultat ohne Überlauf	LD HL,.RES.(7FFFH+1) ; Unterdrückung der Überlaufmeldung

Marken:
- max. 6 Zeichen lange Symbole für 16-Bit- Speicheradressen bzw. Datenwörter
- müssen mit Buchstaben beginnen und dürfen ansonsten nur numerischeund α-num. Zeichen (A-Z), sowie die Zeichen „?" und "_" enthalten

- können in jeder beliebigen Spalte beginnen, sofern sie mit Doppelpunkt abgeschlossen sind
- steht Marke ab Spalte 1, ist der Doppelpunkt nicht nötig

8.4.2.2. Pseudobefehle

Jedes Quellprogramm beginnt im Normalfall mit einer ORG-Pseudoanweisung, welche die Anfangsadresse des Objektprogrammes festlegt (ohne ORG-Anweisung - Speicherplatz NULL). Der Abschluß eines Quellprogramms muß mit einer END-Anweisung festgelegt werden. Folgende Pseudoanweisungen können außerdem verwendet werden:

```
[name:] COND  ausdruck
        :                 ; legt Anfang u. Ende für bedingte
        ENDC  ausdruck      Assemblierung fest

        z.B.:  COND WERT>Ø
               LD A,B
               ENDC
```

[name:] DEFB n ; legt den Hexadezimalwert n auf den nächsten verfügbaren bzw. den durch „name" def. Speicher

z.B.: ZAEHL: DEFB 31H

[name:] DEFL ausdruck ;weist „name" einen Ausdruck zu, wobei sich im Programmablauf die Wertzuweisung ändern kann

[name:] DEFM 'zeichenkette' ;legt den Inhalt der „Zeichenkette" (max. 63 Zeichen) auf den def. Speicherplätzen ab; die Zeichenkette muß in Hochkommata eingeschlossen werden

z.B.: E21: DEFM 'ERR21' (45 52 52 32 31)

[name:] DEFS nn ;Speicherreservierung, wobei momentaner Wert des Adreßzählers um die durch nn angegebene Anzahl Bytes erhöht wird

z.B.: DEFS 5Ø

[name:] DEFT 'zeichenkette'; legt den Inhalt der „Zeichenkette" wie bei DEFM ab und ermittelt zusätzlich auf dem 1. Byte die Länge der Zeichenkette

z.B.: E22: DEFT 'ERR22' (Ø5 45 52 52 32 32)

└ 1= 5 Byte

```
[name:] DEFW nn     ;legt den Hexadezimalwert nn auf den dem
                     momentanen Wert des Adreßzählers entspre-
                     chenden Byte und Folgebyte bzw. auf die
                     durch ‚name' definierte 16 -Bit- Speicher-
                     konstante

name: EQU ausdruck  ; weist einer Marke bzw. Konstanten
                     ‚name' einen bestimmten Ausdruck zu,
                     der sich während des Programmab-
                     laufes nicht ändern darf

                z.B.: WERT: EQU 0
```

Definition globaler und externer Symbole

```
GLOBAL name1, name2, ...   ;auf die unter name1 ... def.
                            Symbole eines Moduls kann auch
                            aus anderen Modulen zugegriffen
                            werden, sofern sie dort mit
                            einer ‚EXTERNAL'-Definition
                            festgelegt worden sind

EXTERNAL name1, name2, ... ;mit den unter name1 ... def.
                            Symbolen erhält man deren
                            Zugriff in anderen Modulen,
                            sofern sie dort unter ‚GLOBAL'
                            festgelegt worden sind
```

8.4.2.3. Makros

Für die Programmerarbeitung immer wiederkehrender Programmabschnitte gibt es zwei Möglichkeiten:

- Makros und
- Unterprogramme

Sowohl Makros als auch Unterprogramme werden nur einmal geschrieben, können jedoch über Aufrufe beliebig oft abgearbeitet werden. Der Unterschied zwischen beiden Methoden besteht darin, daß im Assemblerlauf jedes Unterprogramm als separater Bestandteil übersetzt wird, Makros dagegen in ihrer ganzen Länge bei jedem Aufruf ins Programm eingefügt werden.

Makrodefinition

```
name: MACRO ♦PØ, ♦P1, ..., ♦Pn
[name:] ENDM
```

In Makro-Pseudoanweisung wird dem Symbol "name" ein Makro mit den aktuellen Werten PØ...Pn zugeordnet. Die Gesamtheit der Parameter, die zwischen „MACRO" und „ENDM" stehen, wird als Modellanweisung bezeichnet.

```
z.B.:  LOESCH: MACRO ♦ADR ♦ ZAEHL ♦ WERT
               LD B, ♦ZAEHL
               LD HL, ♦ADR
       A1:     LD (HL), ♦WERT
               INC HL
               DJNZ A1
               ENDM
```

Anwendungshinweise

- beliebige Anzahl der Parameter erlaubt
- „♦" muß vor jedem Parameter stehen und darf in Makros sonst nicht verwendet werden
- "name" darf die Zeichen: BLANK, KOMMA, SEMIKOLON und NUMMERNZEICHEN nicht enthalten
- Parameternamen erscheinen nicht in Symboltabelle
- Makrodefinition muß vor Makroaufruf erfolgen
- Makroaufrufe innerhalb eines Makros sind erlaubt
- Makrodefinitionen innerhalb einer Makrodefinition sind nicht zulässig
- Parameternamen dürfen an jeder beliebigen Stelle der Modellanweisung auftreten

Bei der Assemblierung werden die Makros zunächst in einer Makrozwischendatei ohne Übersetzung abgelegt. Erst mit dem Makroaufruf erfolgt eine Zuordnung der aktuellen Werte und die Übersetzung des angesprochenen Makros in den Objektcode.

Makroaufruf

```
name   'SØ', 'S1', ..., 'Sn'
       ;SØ bis Sn stellen die aktuellen Werte für die
        Formalparameter PØ bis P1 dar, „name" ist der
        Makroname

z.B.:  LOESCH BER3  1ØØ Ø   ; Aufruf des Makros „LOESCH"
       LD B,1ØØ
       LD HL,BER3
A1:    LD (HL),Ø            ; Einfügen des Makros in
       INC HL                 das Objektprogramm
       DJNZ A1
       ENDM
```

- die aktuellen Parameter können beliebige Zeichenfolgen sein, getrennt durch Kommata oder Leerzeichen
- die Zuweisung der aktuellen Parameter beim Aufruf entspricht der durch die Formalparameter PØ bis Pn vorgegebenen Reihenfolge
- entspricht die Anzahl der aktuellen Parameter nicht der Anzahl Formalparameter, so erhalten die fehlenden den Wert Ø

z.B.:

```
TRANS:   MACRO #ADR1 #ADR2 #LAEN #REGHL
         COND REGHL > Ø
         PUSH HL
         ENDC
         LD DE,#ADR1
         LD HL,#ADR2
         LD BC,#LAEN
         LDIR
         COND#REGHL > Ø
         POP HL
         ENDC
         ENDM
```

Makroaufruf a)

```
         TRANS BER1 BER2  1ØØ  1
         COND 1>Ø
x)       PUSH HL
         ENDC
x)       LD DE,BER1
x)       LD HL,BER2
x)       LD BC,1ØØ
x)       LDIR
         COND 1>Ø
x)       POP HL
         ENDC
         ENDM
```

Makroaufruf b)

```
         TRANS BER1 BER2  1ØØ          #REGHLnicht angeführt
         COND Ø>Ø
         PUSH HL
         ENDC
x)       LD DE,BER1
x)       LD HL,BER2
x)       LD BC,1ØØ
x)       LDIR
         COND Ø>Ø
         POP HL
         ENDC
         ENDM
```

x) angekreuzte Befehle werden übersetzt

Symbolgenerator

Der Formalparameter „#$YM" kann in einer Modellanweisung an jeder beliebigen Stelle implizit angewandt werden. Bei der Übersetzung wird der Parameter durch eine 4-stellige Hexadezimalkonstante ersetzt, die bei jedem neuen Makroaufruf um 1 erhöht wird. Auf diese Weise können verschiedene Marken in unterschiedlichen Aufrufen eines gleichen Makros eingesetzt werden. (Ansonsten führt Mehrfachdefinition in einer Marke zur Fehlermeldung)

```
z.B.:        LOESCH:  MACRO #ADR #ZAEHL #WERT
                        :
             A1#$YM:    LD (HL),#WERT
                        :
                        DJNZ A1#$YM
                        ENDM
```

1. Aufruf:

```
                        LOESCH BER3  1ØØ Ø
                        :
             A1ØØØØ:    LD (HL),Ø
                        :
                        DJNZ A1ØØØØ
                        ENDM
```

2. Aufruf:

```
                        LOESCH BER3  1ØØ Ø
                        :
             A1ØØØ1:    LD (HL),Ø
                        :
                        DJNZ A1ØØØ1
                        ENDM
```

Recursion

Innerhalb eines Makros können Aufrufe für andere Makros erfolgen. Jeder recursive Makroaufruf erzeugt eine weitere Makroexpansion.

```
z.B.:  MACR1:  MACRO #A #B
       SYMB:   DEFL #B
               MACR2 '#A'          ; Recursiv-Makro „MACR2"
               ENDM
```

```
MACR2:  MACRO #A
        COND SYMB>Ø
SYMB:   DEFL SYMB-1
        MACR2 '#A-1'
        #A
        ENDC
        ENDM
```

Makroaufruf:

```
        MACR1 RLA 3
SYMB:       DEFL 3
            MACR2 'RLA'
            COND SYMB>Ø
SYMB:       DEFL SYMB-1
            MACR2 'RLA'
            COND SYMB>Ø
SYMB:       DEFL SYMB-1
            MACR2 'RLA'
            COND SYMB>Ø
SYMB:       DEFL SYMB-1
            MACR2 'RLA'
            COND SYMB>Ø
SYMB:       DEFL SYMB-1
            MACR2 'RLA'
            RLA
            ENDC
            RLA
            ENDC
            RLA
            ENDC
            RLA
            ENDC
```

8.4.2.4. Assembler-Kommandos

Die Assembler-Kommandos stehen im Quellprogramm und dienen zur Steuerung des Listing-Formats.
Sie müssen mit einem Stern "*" beginnen und mindestens die unterstrichenen Buchstaben beinhalten.

* Eject	; Fortsetzung auf neuer Seite
* Headingzeichenfolge	; Zeichenfolge (max. 28 Zeichen) wird als Überschrift über jede neue Seite gedruckt
* Include filename	; Einfügen der Datei "filename" in das Quellprogramm
* List OFF	; Abbruch des Ausdrucks
* List ON	; Fortsetzung des Ausdrucks
* Maclist OFF	; Druck-Unterdrückung des kompletten Makroblocks und Makroaufrufs
* Maclist ON	; Druck der kompletten Makros

8.4.2.5. Abweichungen der Mnemoniks des UDOS - Assemblers von CABS 1520

Grundsätzlich ist zu bemerken, daß für die Schreibweise der Mnemoniks die TGL 26176 (Schaltkreisbeschreibung U880) gilt. Die bei UDOS zu beachtenden Differenzen werden im folgenden dargestellt.

1. Die Möglichkeit, eine über das Registerpaar HL adressierte Speicherstelle symbolisch in den entsprechenden Befehlen durch M darzustellen, ist nicht erlaubt. Es muß dann immer (HL) geschrieben werden.

Beispiele:	TGL 26176	UDOS
	LD r,M	LD r,(HL)
	ADD M	ADD A,(HL)
	SLA M	SLA (HL)
	JMP M	JP (HL)

2. Bei allen bedingten Sprüngen (Jump), Unterprogrammaufrufen (Call) und Rückkehrsprüngen vom Unterprogramm (Ret) ist die Bedingung nicht Bestandteil der Mnemoniks, sondern steht im Operandenfeld vor dem Sprungziel (durch Komma getrennt).
Damit ist das Mnemonik für einen beliebigen Sprung immer "JP", bei Relativsprung "JR", für einen Unterprogrammaufruf immer "CALL" und für Rückkehr vom Unterprogramm immer "RET".

Beispiele:

TGL 26176		UDOS	
JMP	nn	JP	nn
JPNZ	nn	JP	NZ,nn
JPPO	nn	JP	PO,nn
JRZ	e	JR	Z ,e
CANC	nn	CALL	NC,nn
CAZ	nn	CALL	Z,nn
CAM	nn	CALL	M,nn
RC		RET	C
RPE		RET	PE
RP		RET	P

3. Sonderfälle:

TGL 26176		UDOS	
EXAF		EX	AF,AF'
IMØ		IM	Ø
IM1		IM	1
IM2		IM	2
IN	n	IN	A,(n)
IN	r	IN	r,(C)
OUT	n	OUT	(n),A
OUT	r	OUT	(c),r
CMP	nn	CP	nn

4. Alle Befehle mit IXH, IXL, IYH, IYL sowie die Befehlsgruppe SLI und der Befehl INF, die in der CABS-Sprachbeschreibung des VEB Robotron - Buchungsmaschinenwerk Karl-Marx-Stadt (Stand: März 1981) ausgewiesen sind, aber lt. TGL 26176 vom Hersteller des U880 nicht garantiert werden, werden auch vom UDOS-Assembler nicht übersetzt.

Beispiele:

```
ADD   IXH
LD    IYL,n
SLI
INF
```

8.5. Linker

8.5.1. Funktion

Der Linker verarbeitet einen oder mehrere Objektmoduln, die durch einen Assembler oder einen anderen Sprachübersetzer erstellt wurden, und gibt ein einzelnes Programm in Form einer ausführbaren Prozedurdatei aus. Der Linker besorgt die Umwandlung der Moduln und löst die Beziehungen zwischen den separat assemblierten Moduln auf. Zusätzlich kann eine Ladelistendatei und eine binäre Symboltabelle erstellt werden.

Der Linker arbeitet mit dem UDOS-Betriebssystem zusammen und ermöglicht damit dem Anwender die Steuerung aller Ein-/Ausgabemöglichkeiten.

8.5.2. Anwendung des Linkers

Der Linker wird vom Betriebssystem UDOS durch das Kommando LINK, gefolgt von einer Liste der Modulnamen und Optionen aufgerufen.

LINK Dateiname^{x} [(Optionen)]

Dabei bedeutet:

^{x} es können ein oder mehrere Dateinamen angegeben werden

[ ] die Angabe in Klammern ist optional

Das Kommando wird beendet entweder durch Betätigen der ET1-Taste oder Semikolon.

Wenn aufgerufen, schreibt der Linker seine Identifikation mit Versionsnummer (n.n) auf die Konsole.

LINK n.n

Der Linker führt zwei Pässe aus. Durch den ersten Paß baut der Linker das Bindeverzeichnis, das die Zuweisung von absoluten Adressen und globalen Namen ermöglicht, auf. Externe Namen müssen teilweise gelöst werden (d.h. Zuweisung der absoluten Adresse zu den entsprechenden globalen Namen).

Durch den zweiten Paß wird der Objektcode jedes Moduls verarbeitet, lokale Bezüge werden umgeordnet, externe Bezüge werden komplett aufgelöst und eine Phasendatei und eine Datei der Ladeliste wird erstellt. Ein optionaler dritter Paß wird ausgeführt, wenn eine Symboldatei zu erstellen ist.

Am Ende des Bindelaufes wird der Fehlerzähler auf die Konsole geschrieben:

n ERRORS

n ist die Zahl der mehrfach definierten globalen Symbole und der ungelösten externen Bezügen.
Die letzte Meldung ist

LINK COMPLETE

Der Bindelauf kann durch Eingabe eines "?" unterbrochen werden. Durch erneute Eingabe eines "?" wird die Abarbeitung fortgesetzt.

Wird die ET2 - Taste betätigt, werden alle Dateien geschlossen und das Programm abgebrochen.

Zwei Fehlermeldungen können beim Bindelauf auf der Konsole angezeigt werden und in die Listdatei geschrieben werden.

MULTIPLY - DEFINED GLOBAL IN MODULE:

Symbolname Modulname

Diese Meldung zeigt, daß der Symbolname als global in zwei verschiedenen Moduln definiert war.

UNRESOLVED EXTERNAL IN MODULE:

Symbolname Modulname

Diese Meldung zeigt, daß ein Symbolname, der als EXTERNAL deklariert war, keinen entsprechenden GLOBAL - Name hat.

Mit

1) ✕ = X
2) ✕ = ✕ + X

X ist eine hexadezimale Zahl

werden den Moduln absolute Speicherplätze zugewiesen. Falls ausgelassen, werden die verschieblichen Moduln aufeinanderfolgenden Speicherbereichen zugewiesen.

z.B.: Drei Moduln sind ab der Adresse 4ØØØ und ein Datenpuffermodul ist ab 8ØØØ zu binden:

LINK %= 4ØØØ M1 M2 M3 %= 8ØØØ DATEN

M1, M2 und M3 werden beginnend mit 4ØØØ aufeinanderfolgend gebunden.

Anmerkung:

Beim Binden absoluter Moduln ist % nicht zu verwenden, da diese Moduln ihre eigene Adressinformation besitzen. (bereits bei der Übersetzung erzeugt)

Segmentierung

Der Linker kann auf Wunsch bis zu 16 nicht unmittelbar aufeinanderfolgende Segmente im Speicher generieren, deren Größe stets ein ganzzahliges Vielfaches der Aufzeichnungslänge der Prozedurdatei ist.

Nur - Binde - Moduln

Hierbei werden den Moduln nur Startadressen zugewiesen und EXTERNAL - und GLOBAL - Symbole aufgelöst, aber es wird kein Maschinencode in den Prozedurdateien erzeugt. Moduln, die nur verbunden werden sollen, erhalten ein Minus vor ihren Namen.

z.B. LINK %= 2ØØØ RESIDENT -OVERLAY.2

Der Modul RESIDENT erhält eine Prozedurdatei mit dem Maschinencode. OVERLAY.2 wird nicht in den Maschinencode umgesetzt. Mit Hilfe von Segmentierung und Nur - Binde - Moduln lassen sich mittels einer Überlagerungsstruktur Programme implementieren, die ein gegebenes Speichervolumen überschreiten.

Modulidentifikation

Der Dateiname kann komplett oder teilweise mit Laufwerkname oder Laufwerkspezifikation angegeben werden.

z.B. 1) LINK MODA MODB.OBJ

2) LINK ¤DOS:2/MULTIPLY

Das erste Beispiel zeigt die Verwendung der unvollständigen Dateinamen MODA und MODB.OBJ, die auf dem Master - Laufwerk gefunden werden können.

Das zweite Beispiel zeigt die Verwendung eines vollständigen Dateinamens (MULTIPLY), der im Laufwerk 2 des Gerätes DOS gefunden wird.

Jeder Objektcode muß vom Typ "binär" mit 128 Bytes pro Satz sein. Zusätzlich enthält ein Objektmodul Informationen, die ihn identifizieren. Der Bindelauf wird abgebrochen, wenn ein Modul nicht die korrekte Identifikation enthält.

Der Dateiname jedes Objektmoduls muß mit dem Anhang ".OBJ"in Groß- oder Kleinbuchstaben enden. Ist dieser Anhang in der Kommandozeile nicht angegeben, ergänzt der Linker automatisch den Anhang, bevor die Datei eröffnet wird.

Die Ausgabedatei des Bindelaufes erhält den Namen der ersten Objektdatei in der Kommandozeile, ohne den Anhang ".OBJ". Die Listdatei und die Symboldatei haben den gleichen Namen wie die Prozedürdatei, allerdings mit dem Anhang .MAP bzw. .SYM.

z.B.: 1 LINK PROG

sucht den Objektmodul PROG.OBJ;
erzeugt PROG, PROG.MAP und PROG.SYM

2 LINK mod1.obj mod2 TTTY.INT

sucht die Objektmoduln mod1.obj, mod2.obj
und TTY.INT.OBJ;
erzeugt mod1, mod1.map und mod1.sym

8.5.3. Optionen

Wenn Optionen eingetragen werden, sind sie in Klammern zu setzen. Sie werden in der Kommandozeile nach dem Dateinamen gesetzt. Das Format der Optionen ist ein Schlüsselwort, das von einem Gleichheitszeichen (=) und einem entsprechenden Wert oder Name gefolgt werden kann. Die signifikanten ersten Buchstaben des Schlüsselwortes werden in der nachfolgenden Beschreibung groß geschrieben.

Entry = n

Spezifiziert die Startadresse für die Abarbeitung der Prozedurdatei. n kann ein hexadezimaler Wert oder ein GLOBAL - Name sein. Wenn die Option nicht angegeben ist, wird die Anfangsadresse des ersten Objektmoduls als Startadresse genommen. Falls ein GLOBAL - Name spezifiziert ist, aber nicht gefunden wird, wird eine Fehlermeldung ausgegeben und die Option wird wie nicht angegeben ausgeführt.

LET

Die vollständige Ausführung des Bindelaufes wird auch bei Fehler in den Symboldefinitionen erzwungen. Wird die Option nicht angegeben, führen Fehler zur Unterbrechung der Erstellung der Phasendatei.

Map = Dateiname

Erzeugt eine Liste mit den Modul - Startadressen und - Rängen, alphabetisch geordneten GLOBAL's mit den entsprechenden Adressen und den Moduln (Ladeliste). Der Dateiname spezifiziert eine Datei, in der die Ladeliste und die Fehlermeldungen (keine Abbruchfehler) ausgegeben werden. Ist die Option ausgelassen, wird die Ladeliste mit dem Anhang ".MAP" und dem gleichen Dateinamen wie die Phasendatei erzeugt. (ISO - Typ)

Name = Dateiname

Weist der Prozedurdatei den angegebenen Namen zu. Sonst entspricht dieser dem ersten Modulnamen ohne den Anhang ".OBJ".

NOMap

Unterdrückt die Erstellung der Datei mit der Ladeliste. Sonst wird eine Ladeliste ausgegeben.

NOWarning

Unterdrückt die Ausgabe von Meldungen über mehrfach definierte GLOBAL's oder ungelöste EXTERNAL's.

Print

Bewirkt die Ausgabe der Ladeliste und Fehlermitteilungen auf dem logischen Gerät SYSLST. Sonst werden Fehlermeldungen auf der Konsole und nicht auf SYSLST ausgegeben.

RLength = n

Spezifiziert die Aufzeichnungslänge einer Prozedurdatei, n ist hexadezimal 8Ø, 1ØØ, 2ØØ, 4ØØ, 8ØØ. Fehlt n, wird n = 8Ø gesetzt.

STacksize = n

Gibt die Größe des Stacks für die Abarbeitung der Phasendatei an. n ist die hexadezimale Zahl der Bytes. Sonst wird n = 8Ø gesetzt.

SYmbol

SYmbol = Dateiname

Erzeugt eine binäre Symboldatei für die absoluten GLOBAL's und internen Symbole jedes Objektmoduls für die Anwendung der symbolischen Fehlersuchprogramme. Wenn der Dateiname nicht angegeben ist, erhält die Datei den gleichen Namen wie die Prozedurdatei mit dem Anhang ".SYM". Ist die Option ausgelassen, wird keine Symboldatei erzeugt.

Anmerkung zu folgenden Wechselwirkungen:

- NOM geht vor MAP = Dateiname
- NOM schließt LNT ein

Falls eine Prozedurdatei nicht erfolgreich generiert wurde, wird keine Symboltabellendatei erstellt und die Listdatei enthält nur die Fehlermeldungen.

Wenn der Anwender eine Arbeitsdatei (d.h. ein Dateiname mit der Länge Ø) als Ausgabedatei spezifiziert, wird diese Datei nach dem Bindelauf gelöscht.

Beispiele:

1) LINK N=1ØØØ PROG 1 PROG 2

   erstellt eine Prozedurdatei: PROG 1 (mit der Startadresse 1ØØØ) und eine Listendatei PROG 1. MAP dem selben Laufwerk wie der Objektmodul PROG 1. OBJ und schreibt keine Ladeliste auf SYSLST.

2) LINK PROG 1 (N =MYDOS:2/PROG.RUN SY NOM P E = MAIN)

   erstellt eine Phasendatei PROG.RUN (mit der Startadresse, die gleich der zugewiesenen

Adresse des GLOBAL's MAIN ist) und eine Symboldatei PROG.RUN.SYM auf Laufwerk 2 des Gerätes MYDOS. Es wird keine Listdatei erzeugt aber es wird die Ladeliste und jede Fehlermeldung auf SYSLST geschrieben.

Definition der logischen Geräte:

Der Linker verwendet folgende logischen E/A-Geräte:

2	CONOUT	Fehlermeldungen
3	SYSLST	Ausgabe der Print - Option
4		Objektdateien
5		Phasendateien
6		Listdatei
7		Symboldatei

Meldungen über Abbruchfehler:

Es gibt verschiedene Bedingungen, die die weitere Abarbeitung unterbrechen. Alle Dateien werden dann geschlossen und das Programm abgebrochen. Die folgende Meldung kann auf der Konsole erscheinen.

INVALID OPTION: s

Dies zeigt an, daß die Zeichenkette s keine gültige Kommandozeilenoption ist.

LINK DIRECTORY OVERFLOW

Zeigt an, daß für die Ausführung des Bindelaufes der verfügbare Speicherbereich nicht ausreichend ist. Das Bindeverzeichnis ist grundsätzlich aus GLOBAL's und EXTERNAL's für jeden Objektmodul zusammengesetzt. Dies benutzt der Linker zum Verschieben und Auflösen von Objektcodebeziehungen. Zusätzlich sind Puffer für die Arbeit mit den Dateien erforderlich. Der Bereich für das Bindeverzeichnis kann klein gehalten werden, wenn die Zahl und die Länge der globalen Symbole reduziert und die Zahl der EXTERNAL's minimiert werden. Diese Meldung erscheint auch, wenn mehr als 127 Moduln zusammengebunden werden.

PROGRAM TOO BIG

Zeigt an, daß der aktuelle Adreßzeiger über FFFFH erhöht wurde. Der Programmbereich oder die Anfangsadresse sollte reduziert werden.

TOO MANY SEGMENTS

Zeigt an, daß mehr als 16 Segmente erzeugt werden sollen. Dieser Fehler kann eintreten, wenn der aktuelle Adreßzeiger oft verringert wird, um ein neues Segment zu erzeugen.

FILE NOT FOUND: dateiname

Zeigt an, daß das Objekt "dateiname" nicht im Verzeichnis gefunden wurde.

INVALID FILE: dateiname

Zeigt an, daß die Zeichenkette "dateiname" kein korrekter Dateiname ist.

INVALID FORMAT: dateiname

Zeigt an, daß das Objekt "dateiname" andere Attribute hat als Typ binär und Satzlänge 128 Byte.

INVALID DATA: dateiname

Zeigt an, daß das Objekt "dateiname" ungültige Daten enthält, wie CRC - Zeichenfehler oder einen Symbolnamen, dessen Länge entweder null oder größer als der maximal zulässige Wert (127) ist. Durch erneute Assemblierung sollte versucht werden, den Fehler zu beheben.

I/O ERROR e ON UNIT u

Alle anderen Abbruchfehler entsprechen I/O - Fehlern (hexadezimaler Wert e) auf einem logischen Gerät (dezimaler Wert u)

8.5.4. Listenformat

Die zugewiesenen Anfangsadressen und die Länge jedes Moduls sowie eine alphabetische Liste aller GLOBAL's mit den zugehörigen Adressen und den Moduln, die sie enthalten, werden in einer vom Linker erzeugten Ladeliste abgelegt.

Zusätzlich enthält die Liste den Programmnamen, seine Länge und die Startadresse.

Beispiel:

```
LINK    X = 4000  MODULA  MODULB
        X = 5000  TAN.MATH  COT.MATH
            (N = VERARBEITUNG    E = BEGINN)

LINK 1.0

LOAD MAP

MODULE          ORIGIN          LENGTH

MODULA          4000            0A73
MODULB          4A73            0319
TAN.MATH        5000            00A3
COT.MATH        50A3            00C5

GLOBAL          ADDRES          MODULE

TAN             5000            TAN.MATH
BEGINN          4085            MODULA
COT             50A3            COT.MATH
SQRT            4A73            MODULB
VEKTOR          45CD            MODULA

PROGRAM VERARBEITUNG - - 1168 BYTES
ENTRY:  4085
```

8.5.5. Überlagerungen

Dieser Abschnitt gibt einige Hinweise für den Anwender, der sein Programm als Überlagerungsstruktur implementieren muß. Die Verwaltung der Überlagerungsstruktur muß durch das Anwenderprogramm realisiert werden. Dabei ist zu beachten, daß das Laden eines Überlagerungssegments erfolgen muß, bevor zu diesem Segment Beziehungen hergestellt werden.

Eine Überlagerungsstruktur besitzt ein residentes Segment, das die Daten und Routinen enthält, die gemeinsam von den verschiedenen Segmenten benötigt werden. Das heißt, die Routine, durch die ein anderes Segment geladen wird, wird sich normalerweise in dem residenten Segment befinden.

Das Problem einer Überlagerungsstruktur läßt sich an einem Beispiel darstellen.

Ein Programm (BEISPIEL) liest eine große Anzahl von Werten von einer Quelle (LESEN), reorganisiert und berechnet neue Daten (VERARBEITEN) und gibt diese Daten wieder aus (SCHREIBEN). Es wird angenommen, daß der verfügbare Speicherbereich nur für ein Segment und die gemeinsamen Routinen ausreicht. Wenn Daten und Routinen als GLOBAL und EXTERNAL in jeden der vier Moduln, die das Programm enthalten, deklariert sind, kann der Anwender diese in einzelnen Bindeläufen wie folgt kombinieren:

Das Segment LESEN paßt zusammen mit dem residenten Segment BEISPIEL in den Speicher. Die beiden werden zusammen zu einer Phasendatei gebunden.

Es ist anzumerken, daß der residente Modul Bezüge zu den Segmenten VERARBEITEN und SCHREIBEN enthält. Diese Moduln werden als "Nur - Bindemoduln" verarbeitet um die Auflösung der Bezüge zu ermöglichen.

```
1) LINK BEISPIEL    X = 5000  LESEN
                    X = 5000  -VERARBEITEN
                    X = 5000  -SCHREIBEN
```

Die beiden Überlagerungssegmente werden erstellt durch jeweiliges Binden mit dem residenten Modul, wobei der residente Modul als "Nur - Bindemodul" verarbeitet wird.

```
2) LINK - BEISPIEL  X = 5000  VERARBEITEN
                              (N - VERARBEITEN)

   LINK - BEISPIEL  X = 5000  SCHREIBEN
                              (N - SCHREIBEN)
```

Die Phasendateien VERARBEITEN und SCHREIBEN werden durch das Programm selbst während der Abarbeitung geladen.

Anlage 1

Besonderheiten PRG 700 mit UDOS 1526

Die Spezifik der Nutzung der ZRE-Karte K 2521 im PRG 700 (K 2526 bei BC 5120) und die Verwendung einer speziellen Tastatur erforderte einige Änderungen im physischen Teil des Betriebssystems.
Die für den Bediener und Programmierer relevanten Veränderungen beziehen sich primär auf die Bedeutung der Tasten der Tastatur und Aktivierung des bildschirmparallelen Druckers.
Die Kommandos des Debuggers wurden geringfügig eingeschränkt.

1. Tastenänderungen

PRG - Taste	BC - Taste (K 7636)
↵	ET 1
ST	ET 2
TAB	-->
CL	CI (Control)
FW	M
↑	^
FC	nicht benutzt
BA	"
+1	¤
+1 mit Umsch.	&
-1	%
-1 mit Umsch.	-

2. Druckeraktivierung

Zu Gunsten der Universalität wurde die Ansteuerung des bildschirmparallelen Druckers nicht in den EPROM's untergebracht. Ein entsprechender druckspezifischer Modul kann aber nachgeladen und aktiviert werden.
z.B.:
%ACTIVATE ¤FSD

Dieses Kommando führt die Aktivierung des hardwaregestützten FS-Anschlusses durch.

3. Debugger

Da das "J"- Kommando durch das "G"- Kommando mit entsprechender Registereinstellung (I=OFH, SP=ODOOH) ersetzt werden kann, wurde auf dieses verzichtet.
Der insgesamt freigewordene Speicherraum wurde zur Erhöhung der Sicherheit für die Hardwaretestroutinen genutzt.
Sie werden beim Einschalten des Gerätes bzw. Laden des logischen BS durchgeführt und bestehen aus den Komponenten:

a) Prüfsummenkontrolle der EPROM's
bei auftretendem Fehler ==> Meldung: PROM DEFEKT!

b) Test der Steuerhardware der Floppy-Disk-Laufwerke
bei auftretendem Fehler ==> Meldung: LW x DEF.

c) Test des Vorhandenseins des notwendigen RAM bei Systemladen bei auftretenden Fehlern == Meldung: RAM DEF.

d) Test des Vorhandenseins einer Systemdiskette
bei Nichtvorhandensein ==> Meldung: KEINE SYSTEMDISK

Im logischen Teil des Betriebssystemes gibt es keinerlei Veränderungen. Es besteht hier volle Kompatibilität mit dem Bürocomputer A 5120. Jedoch ist auf folgende Besonderheiten bei der Systemwartung und der Programmkopierung noch zu achten:

a) Auf Grund der Anpassung des Systemladers an die jeweiligen Erfordernisse kann eine Systemdiskette das logische BS nur auf dem für sie bestimmten Geräten laden.

b) Das Dienstprogramm "FORMAT" zum Neuformatieren von Disketten ist ebenfalls hardwareabhängig und damit nur auf dem PRG 700 zu benutzen.

c) Das Dienstprogramm "COPY.DISK" zum physischen Kopieren von Disketten ist aufwärtskompatibel zum BC A 5120, das heißt auch dort benutzbar, aber nicht umgekehrt das "COPY.DISK" des BC für das PRG.

Notizen